TAB 16

US006453446B1

(12) **United States Patent**
van Ginneken

(10) **Patent No.: US 6,453,446 B1**
(45) **Date of Patent: Sep. 17, 2002**

(54) **TIMING CLOSURE METHODOLOGY**

(75) Inventor: **Lukas P. P. P. van Ginneken**, San Jose, CA (US)

(73) Assignee: **Magma Design Automation, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/054,379**

(22) Filed: **Apr. 2, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/068,827, filed on Dec. 24, 1997.

(51) **Int. Cl.**[7] **G06F 17/50**
(52) **U.S. Cl.** **716/3**; 716/2; 716/6; 716/9
(58) **Field of Search** 716/19, 8, 6, 9, 716/2, 3; 703/19; 713/503

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,168,455 A 12/1992 Hooper
5,237,514 A 8/1993 Curtin

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP 0 610 626 A2 8/1994

OTHER PUBLICATIONS

"Prioritizing factor for nets in timing–based physical design of VLSI chips," IBM Technical Disclosure Bulletin, vol. 33, No. 6B, Nov. 1, 1990, pp. 1–3.

Jones, D.J.; Goesmann, F.; "Photo–thermoelectric power of a–Si as a function of incident wavelength," Journal of Non–Crystalline Solids, vol. 198–200, No. Part 01, May 1996, pp. 210–213.

K.Keutzer, DAGON: Technology Binding and Local Optimization by DAG Matching, *Proc.of the 24th ACM/IEEE Design Automation Conference*, Miami Beach, Fl (Jun. 1987), pp. 341–347, IEEE Computer Society Press 1987.

Grodstein,J.; Lehman,E.; Harkness,H.; Grundmann,B.; Watanabe,Y., "A Delay Model for Logic Synthesis of Continuously–sized Networks," *Digest Int.Conf. on Computer Aided Design*, pp. 458–462, San Jose, Nov. 5–9, 1995.

Lehman,E.; Watanabe, Y.; Grodstein,J.; Harnkess,H.; Logic Decomposition during Technology Mapping, *Digest Inf-.Conf. on Computer Aided Design*, pp. 264–271, San Jose, Nov. 5–9, 1995.

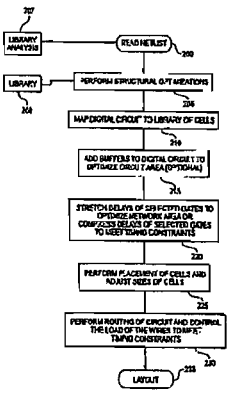
(List continued on next page.)

Primary Examiner—Matthew Smith
Assistant Examiner—Jibreel Speight
(74) *Attorney, Agent, or Firm*—Pillsbury Winthrop LLP

(57) **ABSTRACT**

An automated method for designing an integrated circuit layout using a computer based upon an electronic circuit description and based upon cells which are selected from a cell library, each of the cells having an associated area, comprising the steps of: (a) placing each of the cells in the integrated circuit layout so that the cells can be coupled together by wires to form a circuit path having an associated predetermined delay constraint wherein the cells are coupled together based upon the electronic circuit description input to the computer; (b) connecting the cells together with the wires to form the circuit path; and (c) adjusting an area of at least one of the cells to satisfy the associated predetermined delay constraint of the circuit path.

54 Claims, 14 Drawing Sheets



U.S. PATENT DOCUMENTS

5,282,148	A		1/1994	Poirot et al.
5,392,221	A		2/1995	Donath et al.
5,397,749	A		3/1995	Igarashi
5,402,357	A		3/1995	Schaefer et al.
5,404,312	A		4/1995	Tawada
5,416,718	A		5/1995	Yamazaki 716/5
5,426,591	A		6/1995	Ginetti et al.
5,452,225	A	*	9/1995	Hammer 703/19
5,459,673	A	*	10/1995	Carmean et al. 716/6
5,475,605	A		12/1995	Lin
5,475,607	A		12/1995	Apte et al. 364/489
5,490,268	A		2/1996	Matsunaga
5,508,937	A	*	4/1996	Abato et al. 716/6
5,537,330	A		7/1996	Damiano et al.
5,550,748	A		8/1996	Xiong
5,572,717	A		11/1996	Pedersen
5,581,473	A		12/1996	Rusu et al.
5,619,418	A		4/1997	Blaauw et al. 364/489
5,629,860	A		5/1997	Jones et al.
5,648,911	A		7/1997	Grodstein et al. 716/18
5,654,896	A		8/1997	Roetcisoender et al.
5,657,239	A		8/1997	Grodstein et al.
5,663,889	A		9/1997	Wakita
5,666,290	A		9/1997	Li et al.
5,689,432	A		11/1997	Blaauw et al. 716/18
5,726,902	A		3/1998	Mahmood et al.
5,734,917	A		3/1998	Matsunaga
5,737,236	A	*	4/1998	Maziasz et al. 716/8
5,745,386	A		4/1998	Wile et al.
5,751,593	A		5/1998	Pullela et al.
5,751,596	A		5/1998	Ginetti et al.
5,764,526	A		6/1998	Douceur
5,764,531	A		6/1998	Kojima et al.
5,764,532	A	*	6/1998	Patel 716/9
5,774,371	A		6/1998	Kawakami
5,778,216	A		7/1998	Venkatesh
5,784,600	A	*	7/1998	Doreswamy et al. 713/503
5,798,935	A	*	8/1998	Doreswamy et al. 716/6
5,949,690	A	*	9/1999	Lawman 716/19

OTHER PUBLICATIONS

Sutherland,I.; Sproull,R.; "The theory of Logical Effort: Designing for speed on the back of an envelope," *Advanced Research in VLSI*, pp.3–16, UC Santa Cruz, 1991.

Venkat,K.; "Generalized delay optimization of resistive interconnections through an extension of logical effort," *Proc.Int.Symp.on Circuits and Systems*, 1993, vol.3, pp.2106–2109, Chicago, May 3–6, 1993.

* cited by examiner

FIG. 1

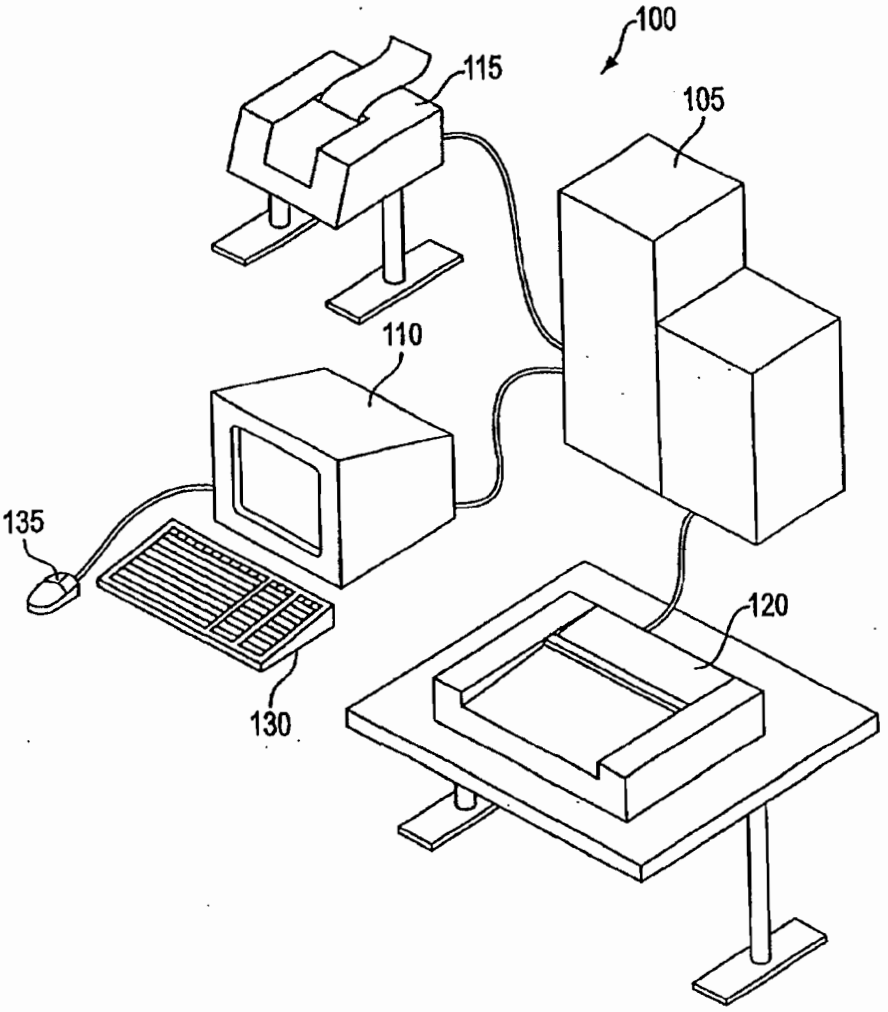


FIG. 2

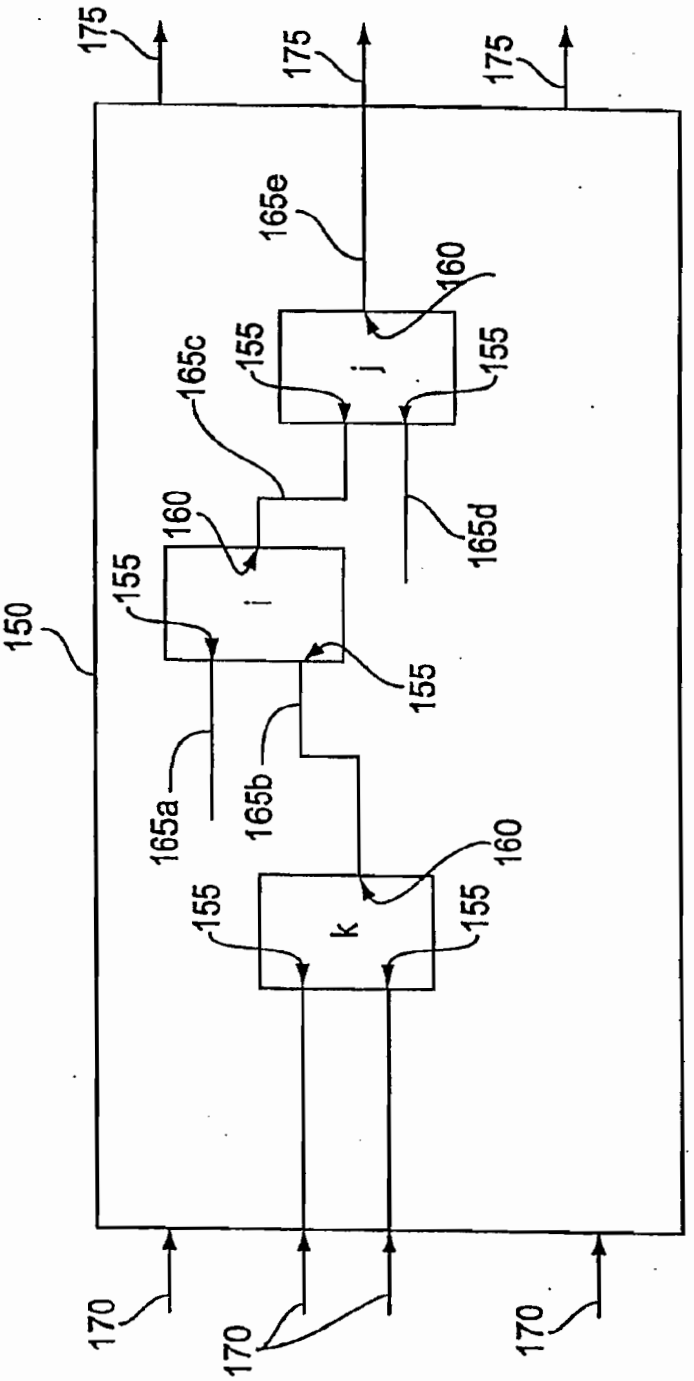

175
175
175
165e
160
155
j
155
165c
165d
160
155
i
155
150
165a
165b
160
155
k
155
170
170
170

FIG. 3

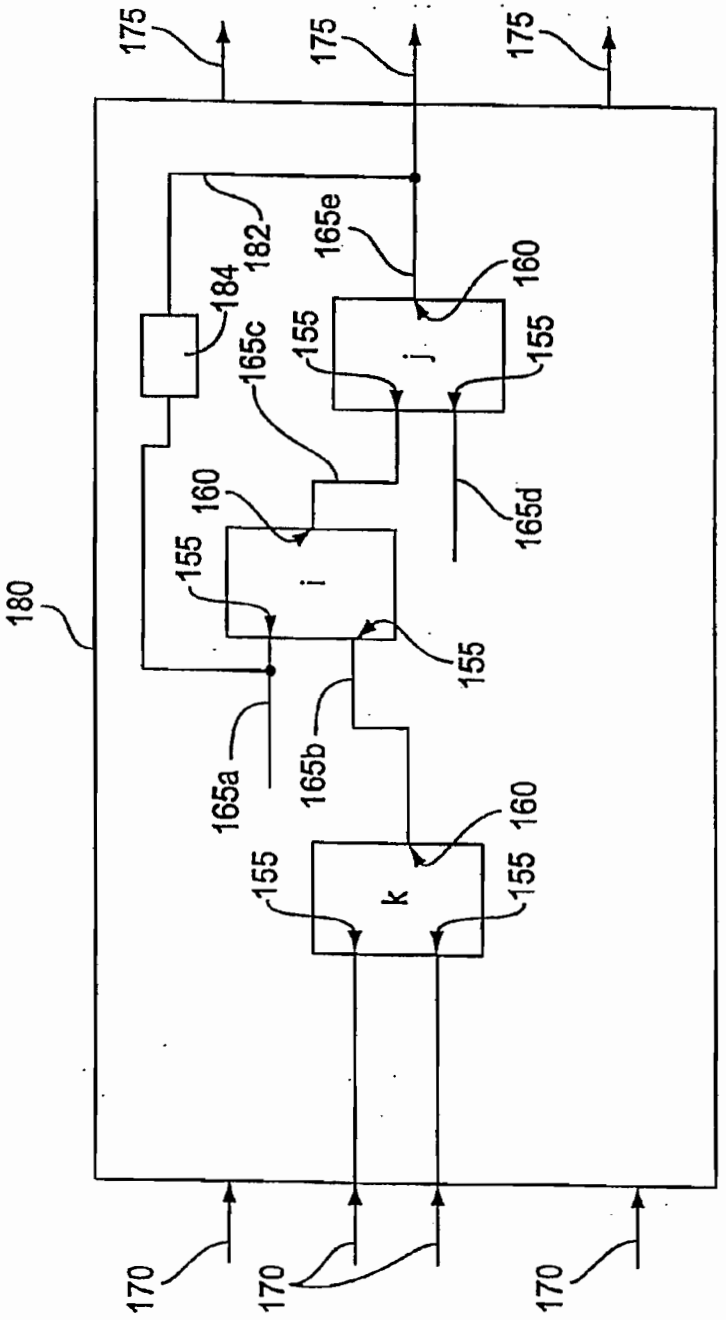
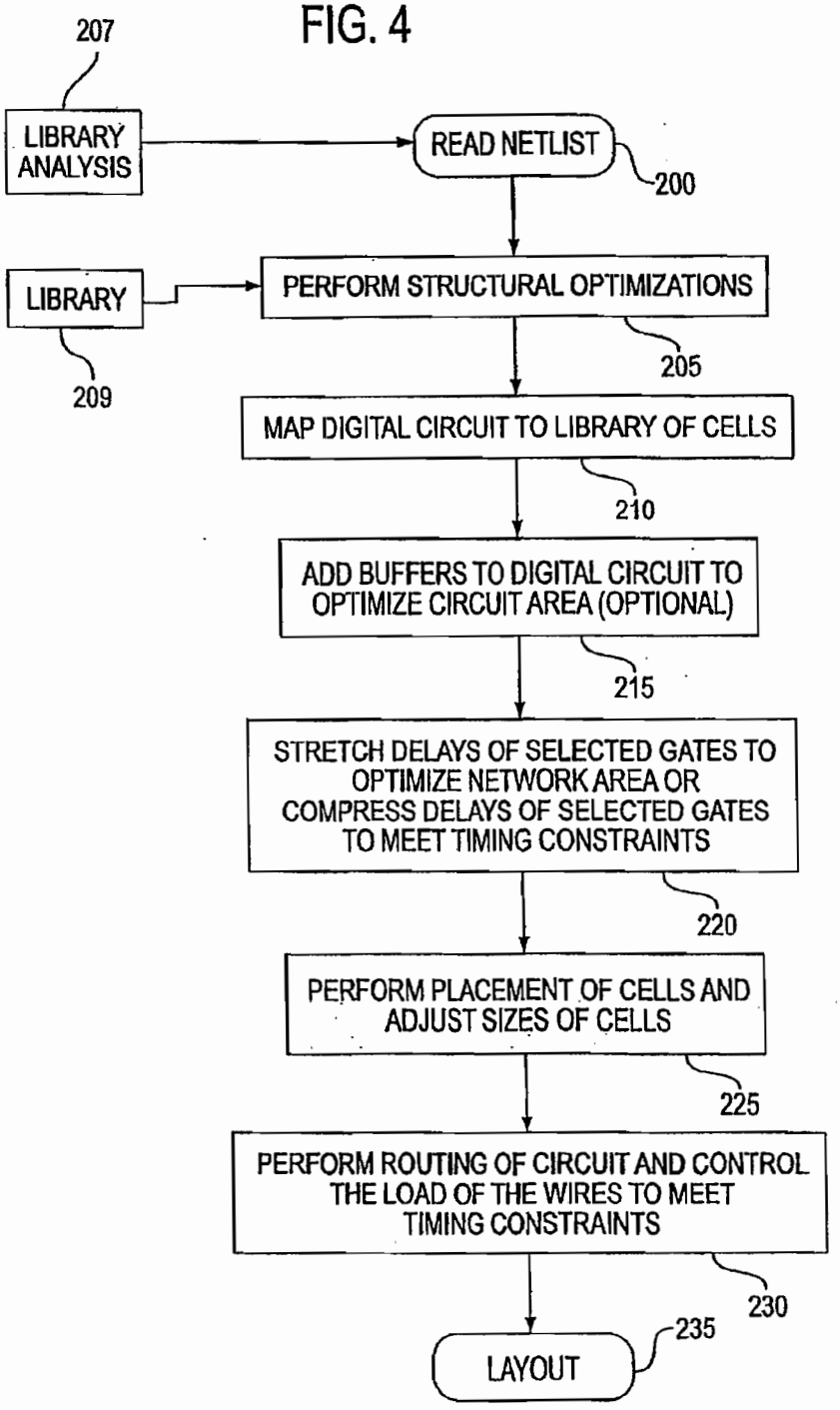

175
175
175
182
165e
184
160
165c
155
j
155
160
155
165d
180
i
155
165a
165b
160
155
k
155
170
170
170

FIG. 4



MAGMA0001210

ATTORNEYS' EYES ONLY

SYN0001345

FIG. 4A

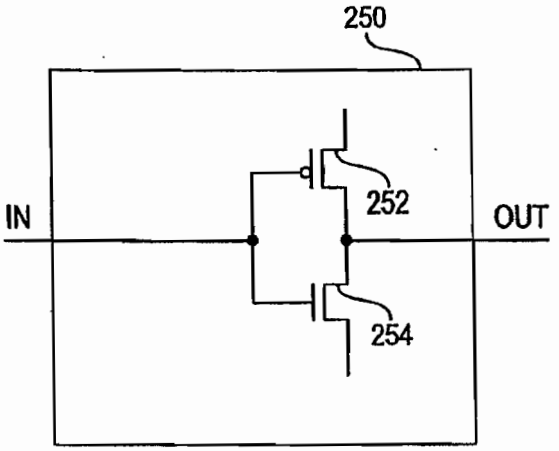


FIG. 4B

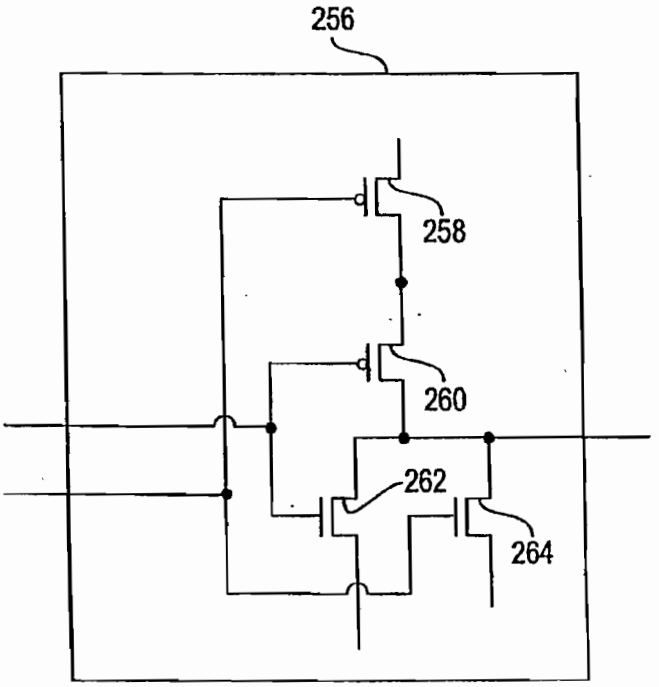
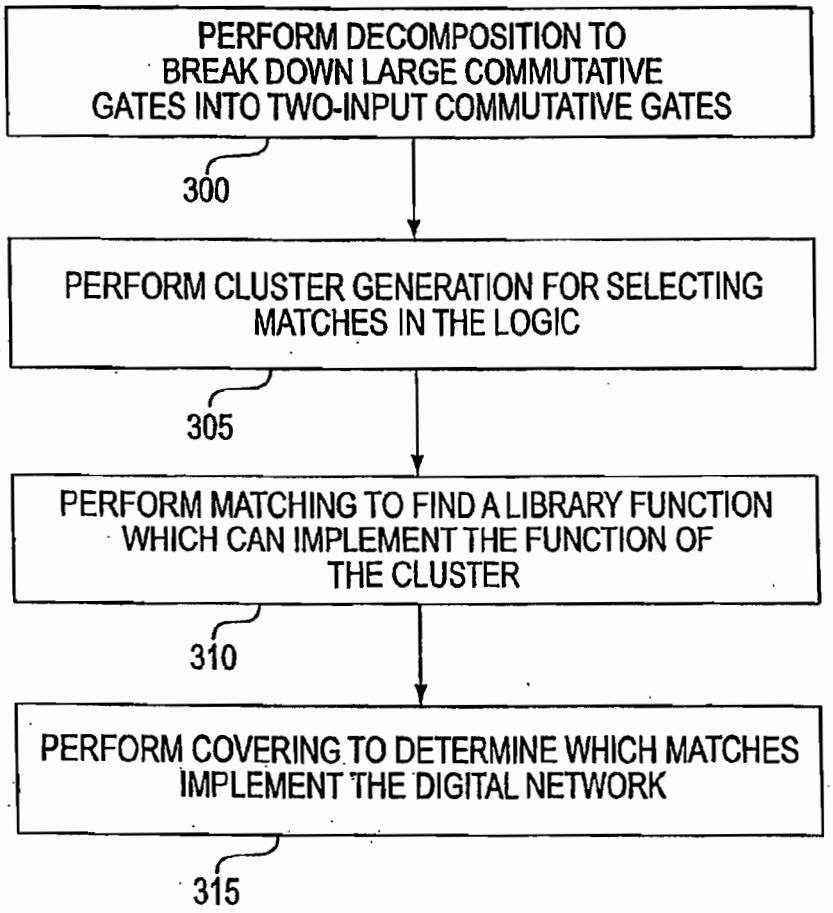


FIG. 5



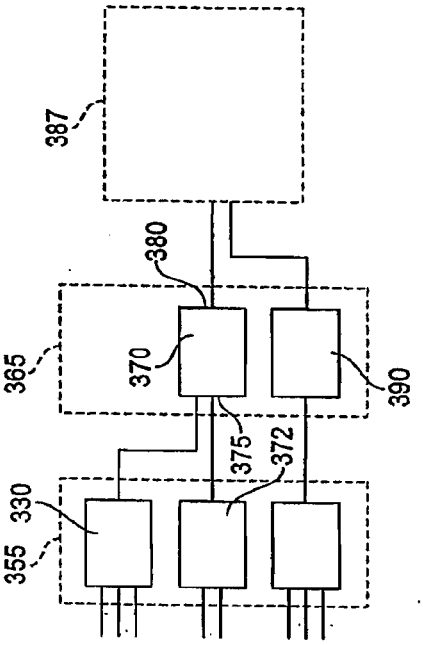


FIG. 6B

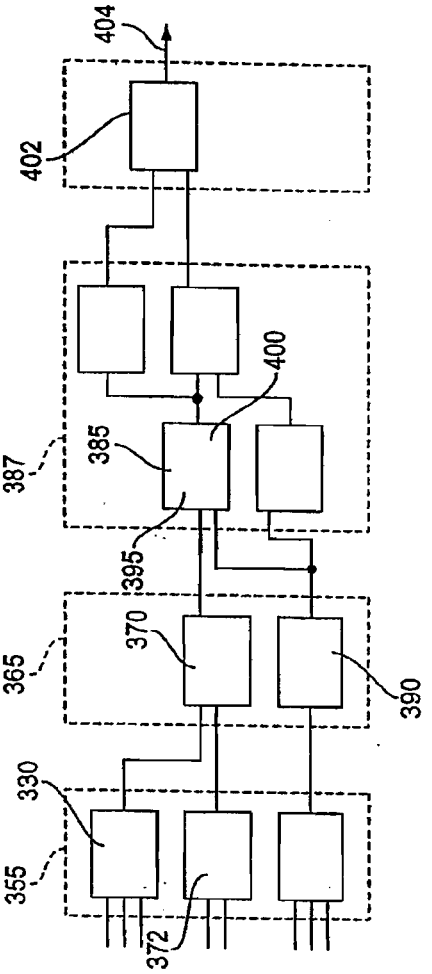


FIG. 6C

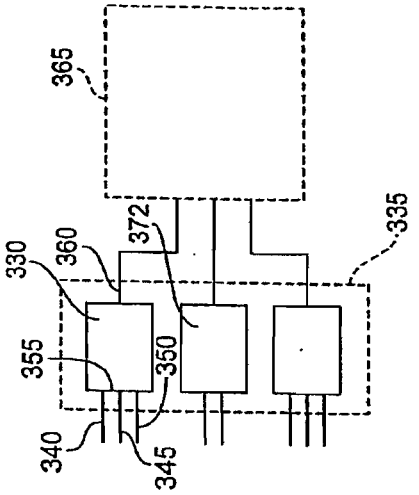


FIG. 6A

FIG. 7A

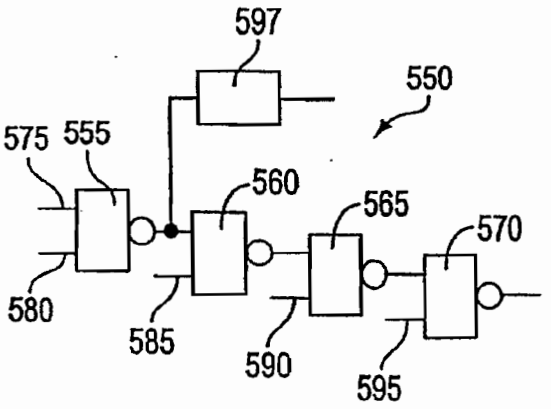

597
550
575
555
560
565
570
580
585
590
595

FIG. 7B

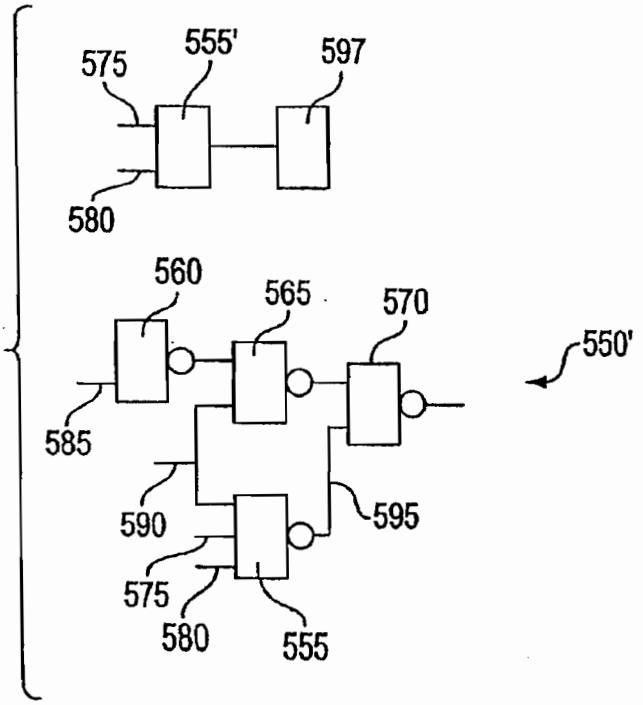
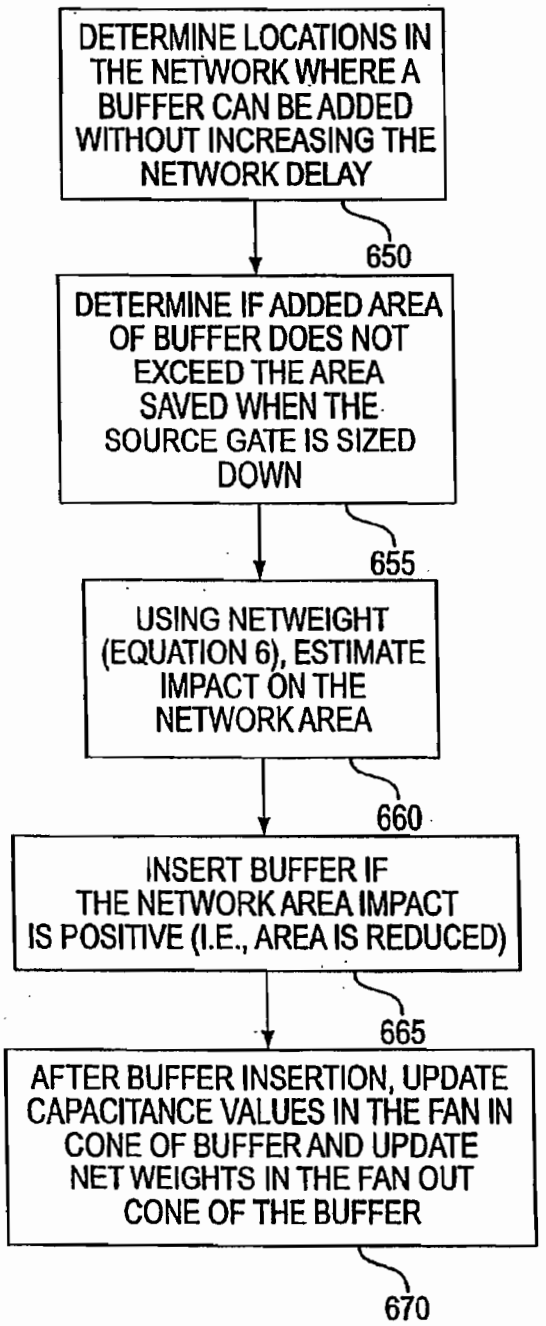

575
555'
597
580
560
565
570
550'
585
590
595
575
580
555

FIG. 8



MAGMA0001215

ATTORNEYS' EYES ONLY

SYN0001350

FIG. 8A

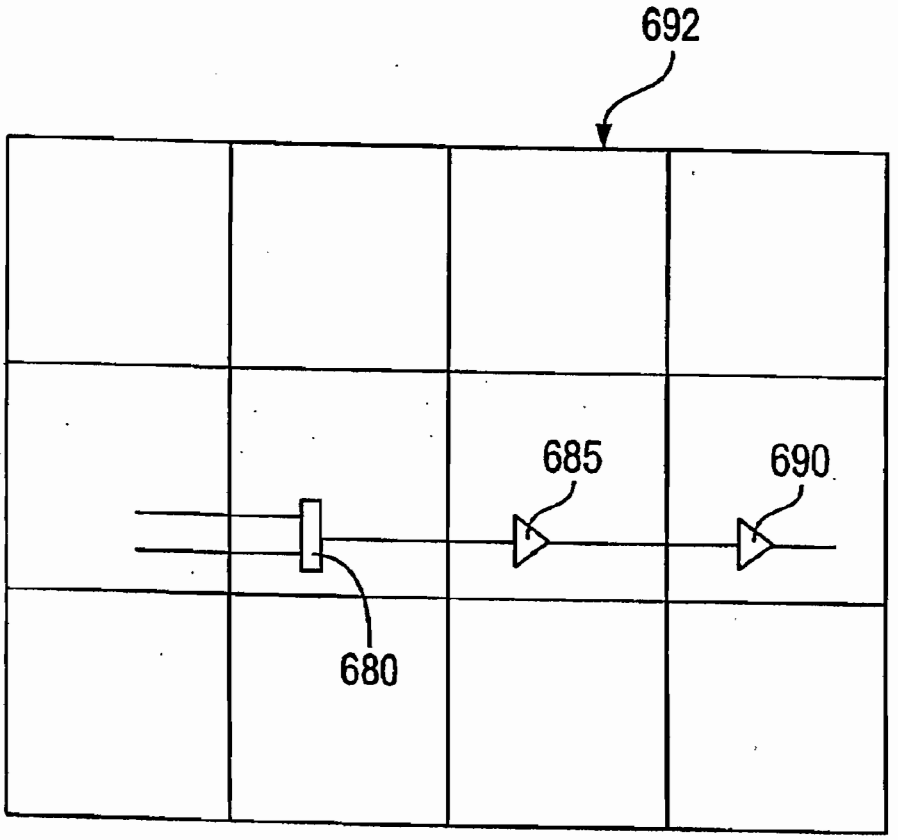


FIG. 9

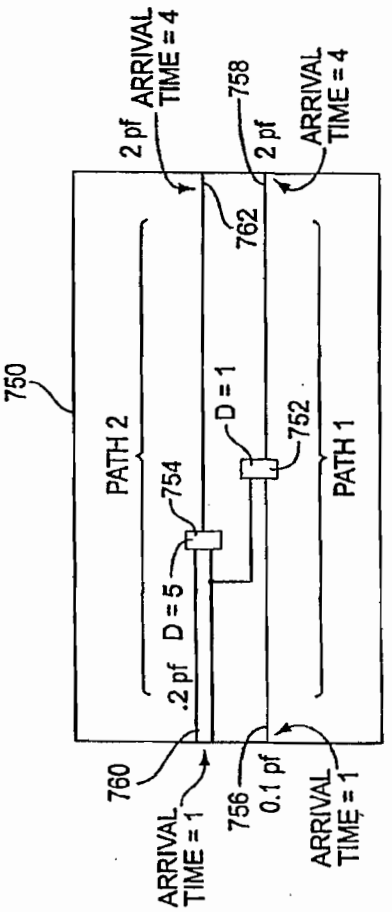

750
PATH 2
PATH 1
754
752
762
758
760
756
2 pf
ARRIVAL TIME = 4
2 pf
ARRIVAL TIME = 4
D = 1
D = 5
.2 pf
0.1 pf
ARRIVAL TIME = 1
ARRIVAL TIME = 1

FIG. 9A

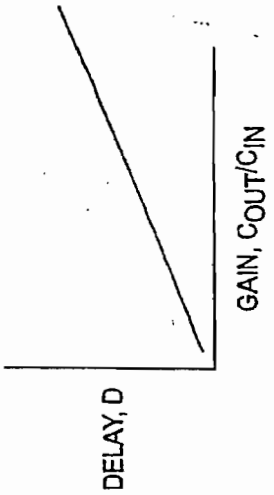

GAIN, C_{OUT}/C_{IN}
DELAY, D

FIG. 9B

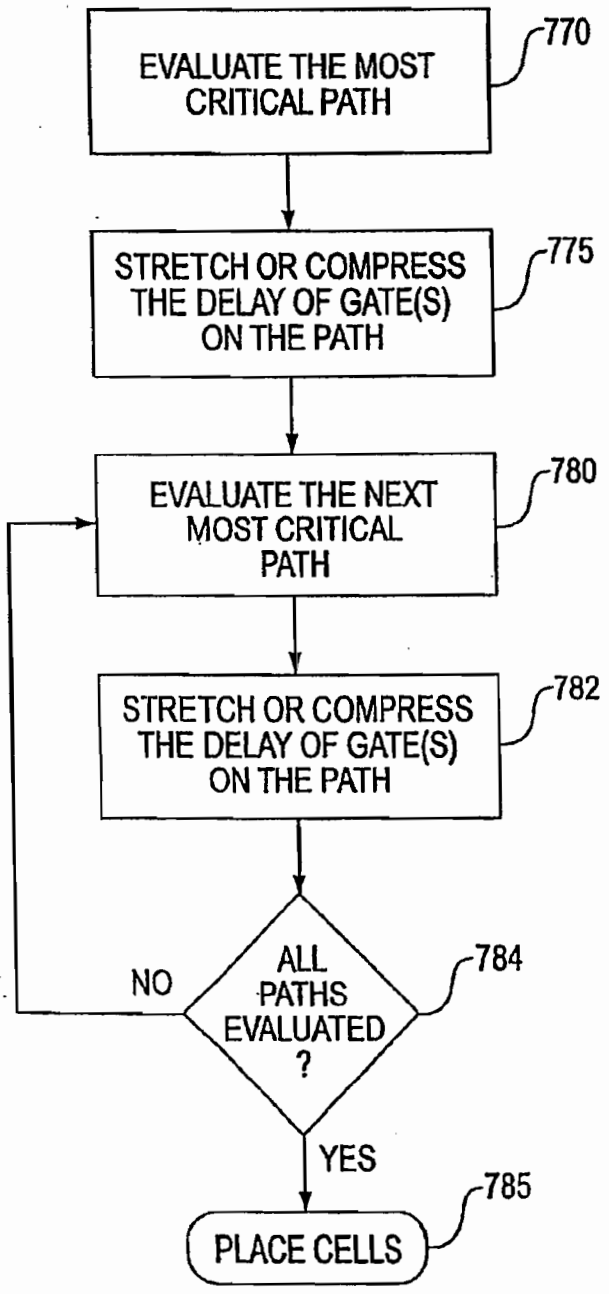

EVALUATE THE MOST CRITICAL PATH
770
STRETCH OR COMPRESS THE DELAY OF GATE(S) ON THE PATH
775
EVALUATE THE NEXT MOST CRITICAL PATH
780
STRETCH OR COMPRESS THE DELAY OF GATE(S) ON THE PATH
782
ALL PATHS EVALUATED ?
784
NO
YES
PLACE CELLS
785

FIG. 10C

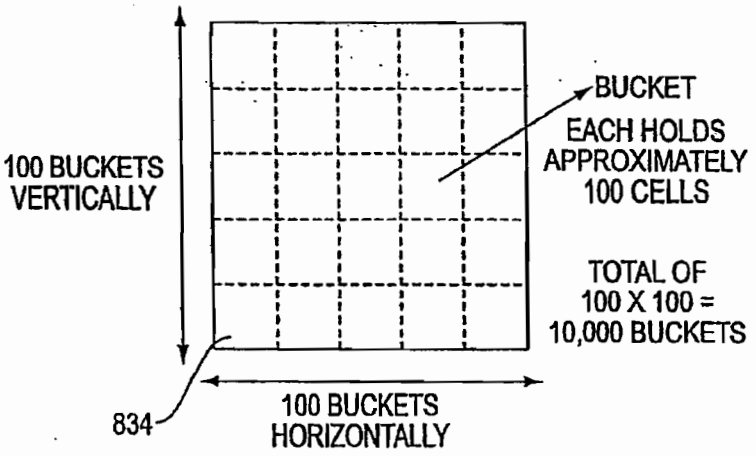


FIG. 10A

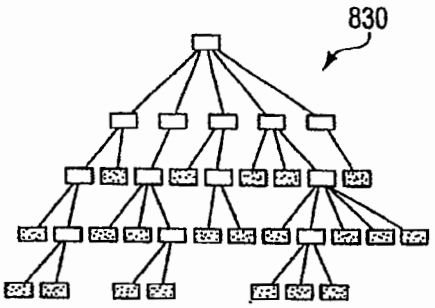


FIG. 10B

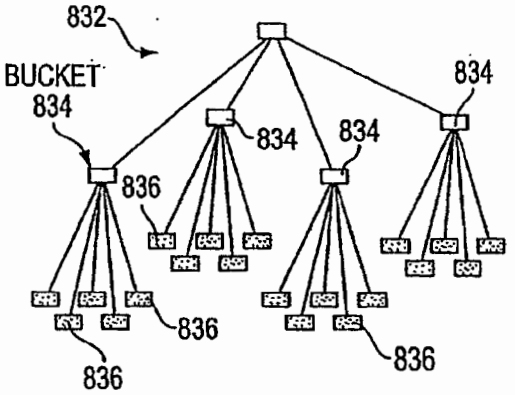
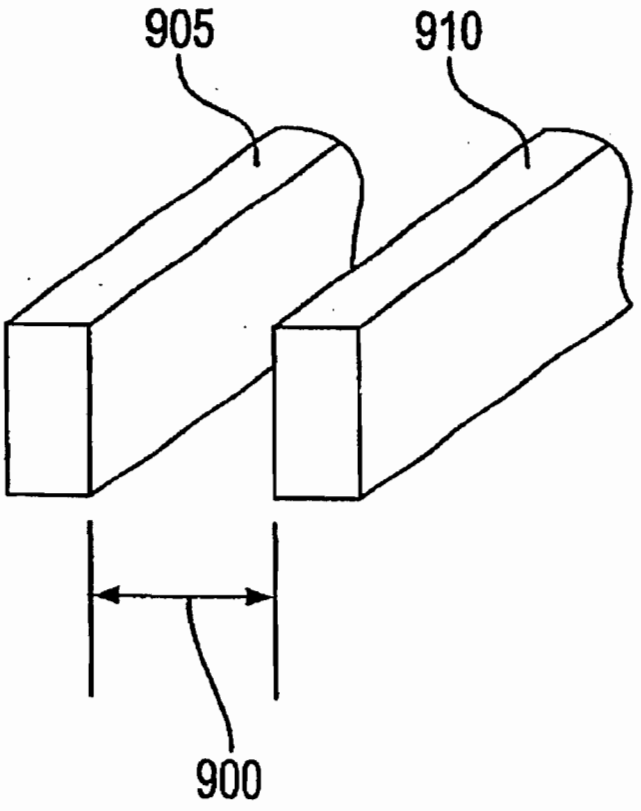


FIG. 11



TIMING CLOSURE METHODOLOGY

REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/068,827 filed Dec. 24, 1997.

FIELD OF THE INVENTION

This invention relates generally to the field of integrated circuit design and more specifically to a methodology for meeting timing constraints in the design of digital circuits.

BACKGROUND OF THE INVENTION

In designing electronic circuits and systems, computer-automated design systems are used for defining and verifying various prototype circuit configurations. As part of the circuit definition, delay constraints are specified by the circuit designer. These delay constraints should be satisfied when the prototype circuit is fabricated.

In conventional approaches to circuit design, the following steps are typically performed:

(1) the load capacitance for each cell in the circuit is estimated using a fanout-based model;

(2) the size of each cell is set to optimize timing of the circuit;

(3) the cells are placed, and the net (wire) lengths of the circuit are estimated;

(4) the wires are routed; and

(5) final analysis is made to determine whether timing closure (i.e., satisfaction of timing constraints) is achieved.

In step (2), the sizes of cells within the circuit are chosen and held constant once chosen. The placement algorithm used thereafter will assign different net lengths between cells, and these lengths have conventionally been difficult to predict prior to placement. While net lengths have been estimated prior to placement by use of an estimation function or table which gives the load value of a net based on the number of fanout gates, this estimation function is usually inaccurate. This difficulty in accurately predicting net lengths leads to unpredictable delay effects after cell placement occurs. For example, some nets turn out to be longer in length than expected. These longer nets cause longer delays which prevent satisfaction of timing constraints in the digital circuit. Thus, under the conventional design approach, timing closure is not certain until after placement.

Failure to achieve timing closure after placement leads to additional expenses and other problems for the designer. To correct for failure to achieve timing closure, the designer has the option of fixing the design manually, which is difficult and time consuming because the automatically optimized digital network is not easy to understand. As a second option, the designer may change the Hardware Description Language (HDL) specification and repeat the design process. However, timing closure will again not be certain until after placement. Thus, the design process must again be repeated before the designer can determine if the HDL specification changes were successful in enabling timing closure.

A common method for dealing with inaccurate net load estimates is by estimating the net load at a considerably larger value than typically estimated. Although this method increases the probability of meeting timing constraints after placement, it causes the sizes of the gates to be considerably larger than necessary. Gates which are larger than the necessary size are wasteful in both silicon area and power consumption. This leads to chips which are larger, more expensive to produce, and use more electrical power than necessary.

Another problem with the conventional circuit design approach concerns the timing analysis required during optimization and during placement. The timing analysis performed throughout the conventional circuit design process is very time consuming, and accounts for most of the run time of a conventional circuit design system.

A further disadvantage of the conventional design approach relates to the net length modifications performed by the placement program. Depending on the location chosen for each gate, each net length may be modified. As each net length is modified, the capacitive load of the net will change. As a result, the delays of the gates driving the net will change. Therefore, the delays, which were carefully optimized during the logic design, are very different in value after cell placement, thereby contributing to poor network optimization.

Additionally, most of the progress in the state of the art for digital circuit design can be characterized as increased integration which has led to increasingly complex software systems which are slow, and difficult to design and maintain.

A further disadvantage with conventional design approaches is in the difficulty of iterating between placement and sizing, since the logic synthesis program is often operated by the logic designer who also wrote the HDL specification, but the placement program is often operated by the silicon chip manufacturer, after the design is complete.

One proposal has been made to keep delay constant while expressing size as a linear function of the gate load. See, J. Grodstein, E. Lehman, H. Harkness, B. Grundmann, and Y. Watanabe, "A Delay Model for Logic Synthesis of Continuously-Sized Networks", Digest Int. Conf. On Computer Aided Design, pp. 458–462, San Jose, Calif. Nov. 5–9, 1995. Under this proposal, as the gate load changes, the gate size automatically grows sufficiently to hold the delay constant.

The constant delay model has been proposed in a mapping algorithm. See, E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic Decomposition During Technology Mapping", Digest Int. Conf. On Computer Aided Design, pp. 264–271, San Jose, Calif. Nov. 5–9, 1995. However, this proposal does not provide a good method for choosing the constant delays and, in addition, it only applies the constant delay model to mapping.

Similarly, the constant delay model is proposed for fanout optimization. See, K. Kodandapani, J. Grodstein, A. Domic, and H. Touati, "A Simple Algorithm For Fanout Optimization Using High Performance Buffer Libraries", Digest Int. Conf. On Computer Aided Design, pp. 466–471, Santa Clara, Nov. 7–11, 1993. While the above-mentioned references note the importance of the constant delay model, they do not note the importance of gain, also termed "electrical effort".

In Sutherland and R. Sproull, "The Theory of Logical Effort: Designing for Speed on the Back of an Envelope", Advanced Research in VLSI, pp. 3–16, University of California, Santa Cruz, 1991, the delay is noted as being dependent on gain. A size independent formulation of the delay optimization problem is also presented, but the solution is intended for use as imprecise, scratch-pad type calculations, and not part of an overall integrated, automated solution to cell placement in the design of integrated circuits. In V. Kumar, "Generalized Delay Optimization of Resistive Interconnections Through an Extension of Logical Effort,

Proc. Int. Symp. On Circuits and Systems, 1993, vol. 3, pp. 2106–2109, Chicago, Ill., May 3–6, 1993, the methods above are used to analyze long wires with a significant amount of RC delay.

In U.S. Pat. No. 5,654,898, a method and apparatus for determining an integrated circuit layout is shown and described whereby timing-driven buffer sizing is performed to satisfy timing requirements. However, this solution also relies on imprecise calculations and does not teach or suggest an overall integrated, automated solution to cell placement in the design of integrated circuits.

What is needed and what has been invented is a method and apparatus for overcoming the foregoing deficiencies, and for maintaining timing closure upon placement and routing of the digital circuit or network.

SUMMARY OF THE INVENTION

The present invention broadly provides a method for designing an integrated circuit layout based upon an electronic circuit description and by using a cell library containing cells that each have an associated relative delay value, comprising the steps of:

(a) selecting a plurality of cells that are intended to be coupled to each other with a plurality of wires and that can be used to implement the digital circuit based on the electronic circuit description;

(b) determining an initial intended location of each of the selected plurality of cells on the integrated circuit, the step of determining the initial intended location of each of the selected plurality of cells including the step of determining an initial intended area of each of the selected plurality of cells, the initial intended area of at least some of the selected plurality of cells being determined using the associated relative delay value of the selected cell and the initial intended lengths of some of the wires coupled to the selected cell;

(c) finalizing the location and area of each of the selected plurality of cells and the lengths of the wires; and

(d) routing the digital circuit to generate the integrated circuit layout using the finalized location and area of each of the selected plurality of cells and the finalized wire lengths.

The present invention makes possible an advantage of facilitating the maintenance of timing closure throughout the design process. The invention makes possible another advantage of compensating for the unpredictable delay effects due to the placement and routing steps in the design process.

Still another advantage made possible by the invention is the avoidance of costly timing analysis of conventional approaches. In contrast, timing analysis is required during placement and throughout the conventional design process.

The following are also made possible by the invention by establishing circuit area, instead of timing, as an optimization objective. First, excess circuit area which is formed in one module can be compensated in another module. In contrast, excess timing in one module cannot generally be compensated by another module. Second, area is a measurable parameter when comparing two different circuits. In contrast, it is difficult to compare two circuits for performance advantages, based on timing, since timing may be different for every circuit path. Third, if circuit area is established as a constraint, then routing completion is not guaranteed to be achieved.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic diagram of a host computer system which is capable of implementing the present invention;

FIG. 2 is a schematic circuit diagram of a combinational network which can be designed by the computer system of FIG. 1;

FIG. 3 is a schematic circuit diagram of a sequential network which can be designed by the computer system of FIG. 1;

FIG. 4 is a flowchart describing a method of the present invention;

FIG. 4A is a schematic diagram of a real, non-ideal inverter;

FIG. 4B is a schematic diagram of a NOR gate;

FIG. 5 is a flowchart describing a method of mapping;

FIG. 6A is a schematic diagram of a portion of a circuit shown for discussion of "arrival" time calculations;

FIG. 6B is a schematic diagram of the circuit of FIG. 6A shown with its next circuit stage;

FIG. 6C is a schematic diagram of the circuit of FIG. 6B shown with its next circuit stage;

FIG. 7A is a schematic diagram of a gate chain with four levels of logic;

FIG. 7B is a schematic diagram of the gate chain in FIG. 7A after transformation wherein the number of levels of logic is reduced;

FIG. 8 is a flowchart describing a method of buffering according to the invention;

FIG. 8A is a schematic diagram of a portion of a digital circuit wherein buffers are inserted to optimize network area;

FIG. 9 is a schematic diagram of a portion of a digital circuit shown for describing a method for stretching and compressing of gate delays;

FIG. 9A is a graph showing the relationship between the gate gain C_{out}/C_{in} and the gate delay D;

FIG. 9B is a flowchart describing a method for stretching and compressing gate delays according to the present invention;

FIG. 10A is a schematic block diagram of a logical hierarchy of a network according to the invention;

FIG. 10B is a schematic block diagram of a physical hierarchy of a network transformed from the logical hierarchy of FIG. 10A;

FIG. 10C is a partial top elevational view of a plurality of "buckets" wherein cells are placed inside the buckets; and

FIG. 11 is a partial perspective view of two wires in a net shown for describing adjustment of capacitive load values to control timing constraints after routing.

DETAILED DESCRIPTION OF THE INVENTION

Those of ordinary skill in the art will realize that the following description of the present invention is illustrative only and not in any way limiting. Other embodiments of the invention will readily suggest themselves to those skilled in the art.

Referring in detail now to the drawings wherein similar parts or steps of the present invention are identified by like reference numerals, there is seen in FIG. 1 a schematic diagram of a host computer system 100 which is capable of implementing the present invention. The host computer system 100 preferably includes a computer 105, a display 110, a printer 115 and a plotter 120. The computer 105 is typically a personal computer or workstation. The display 110 displays both graphical and textual materials relating to

the design of integrated circuits. Also included in the computer system **100** is a keyboard **130** and a pointing device **135** such as a "mouse". When operating with software tools used in integrated circuit design, the computer system **100** essentially becomes a series of electronic circuits for accomplishing specific design functions. One example of such software tools is the Aquarius which is commercially available from Avant! Corporation of Fremont, Calif.

FIG. **2** shows a portion of a typical integrated circuit **150** which can be designed by the computer system **100** (FIG. **1**) and which is provided as additional background material for understanding the present invention. The digital circuit **150** comprises a plurality of gates, including gates i, j, and k. The gates can be combinational gates whose function is represented as Boolean expression based on, for example, the operators AND, OR and NOT. The gates can also be registers.

Each gate (e.g., gate j) has one or more input **155** and a single output **160**. The gates in the circuit **150** are coupled together by a plurality of nets. For example, the gate i has a first input connected to a net **165***a* and a second input connected via net **165***b* to the output of the gate k. The gate j has a first input connected via net **165***c* to the output of gate i, a second input connected to the net **165***d*, and an output connected to the net **165***e*.

Gates whose outputs are connected to the inputs of a gate are collectively called the "fanin" of the latter gate. Thus, the gate k is in the fanin of the gate i. Gates whose inputs are connected to the output of a gate are collectively called the "fanout" of the latter gate. Thus, the gate j is in the fanout of the gate i.

The digital circuit **150** performs a logic function by processing digital binary input data in a number of cycles. The input data is presented to the digital circuit **150** at the primary inputs **170**, and the result of the computation of the digital circuit function is presented at the primary outputs **175**. Typically, the computation of the digital circuit function requires one or more cycles. During each cycle, the gate functions are calculated, and the calculation results are stored in registers for use in the next cycle.

The circuit **150** of FIG. **2** is known as a combinational network. FIG. **3** is a schematic diagram of a sequential network **180** which can also be designed by the computer system **100** (FIG. **1**). In the sequential network **180**, the feedback loops typically include at least one register. For example, the feedback loop **182** includes the register **184**.

FIG. **4** is a flowchart describing an overview of the automated method of the present invention that is implemented using a computer program containing sequences of instructions that implement each of the various steps described hereinafter. The automated program thus implements a method that includes the following steps below. It is noted further that in the flowchart of FIG. **4**, each of the steps has its own distinct computer instructions (i.e., each of the steps **200**–**230** comprises further steps). In step **200**, a netlist generated by synthesis (or other circuit description) is read. In step **205**, structural optimizations are performed to reduce the number of connections in the digital circuit, while maintaining the overall circuit function. Additionally, library analysis **207** of library **209** may be performed during, or prior to, step **205**. As discussed in further detail below, during the library analysis **207**, the delay is determined for each gate to be used in the digital circuit. In step **210**, the digital circuit is mapped to the library **209** of cells. In step **215** buffers may be added as an option to the digital circuit to optimize the circuit area. In step **220**, "stretching" and/or "compressing" of gate delays may be performed to assist in satisfying the predetermined timing constraints and to optimize the network area. In step **225**, cell placement occurs, during which the size of a cell is determined to maintain the constant cell delay characteristics according to the present invention. In step **230**, network routing is performed, and the net load of a cell may be adjusted to further tradeoff between timing constraints and circuit area. After step **230**, the chip layout is generated (step **235**).

Read Netlist

In step **200** (FIG. **4**), the invention will read a netlist which is a detailed interconnection listing of boxes in the target technology from which automated logic diagrams may be produced for integrated circuit fabrication. The netlist is typically in Verilog or VHDL format and is generated from synthesis tools such as the Synopsys Design Compiler™ from Synopsys, Inc. or BuildGates™ from Ambit Design Systems, Inc. The synthesis tool performs parsing of the complex Verilog or VHDL constructs which describe the desired logic function. The synthesis tool further performs sequential optimization such as behavioral synthesis, retiming, resource sharing, scheduling, state machine optimization, and register mapping.

Library Analysis

During, or prior to, step **210** in FIG. **4**, a library analysis **207** is performed on the library **209** which will be used for the circuit design. Library analysis **207** is not dependent on the actual circuit which is being synthesized and thus can be performed separately from the circuit design process shown in FIG. **4**. During the library analysis **207**, the delay D is determined for each gate to be used in the digital circuit. Assume a gate in the library **209** has a capacitive load value C and a size value S wherein S is a continuous, positive real variable which linearly increases both the driving capability and area of the gate. The delay D of a gate can be approximated by equation (1):

$$D=f(C/S) \tag{1}$$

The delay D is non-negative and increases as the C/S value increases.

The delay D value depends on the delay provided by a cell plus the delay provided by the net (wire) load of the cell. Thus, in order to ideally maintain the delay D at a constant value prior to and after placement/routing, the net load resistance must be taken into consideration. According to the invention, the gate size is adjusted after cell placement based on changes in the capacitive load in order to maintain D as "constant," as will be described hereinafter in further detail. However, since the wire load also has resistance, the value of D (after gate sizing) may actually vary by a negligible amount from the pre-determined D value. Since this variation is negligible, D may be approximated as "constant" prior to cell placement and after cell placement and gate size adjustment.

The delay D value may also be different for different inputs of the gate and it may also be different for the falling transition and rising transition of a signal propagating through the gate. The C/S value is defined as the "typical load" since it is the load value which can be driven by a gate of size "1". The library analysis will determine a "good" value for C/S for each gate in the library based on gain considerations. It is known to those skilled in the art that to attain maximum gain in a set of series-connected ideal inverters, the gain of each inverter should be e, which is

about 2.7. However, an actual inverter is non-ideal and therefore has a parasitic self-load which results in a parasitic delay which is fixed and independent of the size of the inverter and of the load capacitance driven by the inverter. The principal contributors to the parasitic delay are the capacitance of the source/drain regions of the transistors and the Miller capacitance between the source/drain and the gate of transistors which drive the output of the inverter. If this parasitic delay is taken into account, then to obtain maximum gain in the series-connected inverters, the gain of each inverter is set at greater than 2.7, for example about 3.6.

Discussion is now turned to the method of finding the preferred gain of gates other than inverters. For any circuit or network, a good operating point is the point at which a good trade-off between gain and delay can be achieved. To optimize this trade-off, the "continuous buffering assumption" is used which is based on the ideas described in I.E. Sutherland and R. F. Sproull, "The Theory of Logical Effort: Designing for Speed on the Back of an Envelope," *Advanced Research in VLSI*, pp. 1–16, UC Santa Cruz, 1991.

The method for finding the constant delay is based on a comparison of the gate to an inverter. The method is based on the idea that, at some point, it becomes more advantageous to use a buffer to increase the gain rather than to increase the gain of the gate itself. By adding more buffers, wherein each added buffer is of a larger size than a previously added buffer, the gain of the combination (gate and buffers) can grow exponentially as delay increases.

In the following analysis, it is assumed that continuous buffering can be made whereby it is possible to insert an infinitesimally small amount of buffering. On a given path with several gates, several fractional buffers can be combined into a single real buffer. Thus, for a circuit with many levels of logic, continuous buffering may be achieved approximately or approaches reality.

Assuming continuous buffering can be made, the delay of a series of ideal inverters can be written as $\tau\log(h_{inv})$ wherein τ is a time constant and h_{inv} is the inverter gain. In other words, the gain of a series of buffers increases exponentially with its delay.

Consideration is now turned to the delay of a gate which is coupled to a series of ideal inverters. As shown in equation (2a), the delay d_{gate} of the gate partly depends on the load of the gate, wherein the load of the gate is the load of the combination (C_{out}) divided by the gain h_{inv} of the inverters.

$$d_{gate}=P_{gate}+R_{gate}(C_{out}/h_{inv})+\tau \log (h_{inv}) \quad (2a)$$

The parameter P_{gate} is the intrinsic delay of a gate, while R_{gate} is the gate resistance.

To minimize d_{gate} for a given C_{out} the derivative of d_{gate} (equation (2a)) is taken with respect to h_{inv}, and the result is shown by equation (2b).

$$\partial d_{gate}/\partial h_{inv}=(R_{gate}C_{out}/h^2_{inv})+(\tau/h_{inv}) \quad (2b)$$

By setting equation (2b) equal to zero (0), equation (2c) is derived.

$$R_{gate}C_{out}=\tau h_{inv} \quad (2c)$$

At the point where a buffer is not needed, $h_{inv}=1$ and $R_{gate}C_{out}=\tau$. By substituting equation (2c) into equation 2(a), the delay of each gate is the intrinsic delay P_{gate} plus the time constant τ.

The interpretation of using this constant delay is that this delay is the cross-over point between using buffers versus stretching for gain. In other words, the delay of a gate on a critical path should never be slower than this cross-over point value. This provides a clear criterion for dealing with gain or delay optimization problems. If more path gain is required, a buffer is a faster solution than slowing down the gate. Also, use of a buffer will be a more area-efficient solution.

These results above are used as follows. For a given library, the inverter is first analyzed. Using the result that the gain of an inverter should be about 3.6, the typical load C_{typ} is derived as shown by equation (2d):

$$C_{typ}=(3.6)C_{in} \quad (2d)$$

wherein C_{in} is the input capacitance of the inverter. The typical load is used to derive the constant delay for the inverter. Since this is a real, non-ideal inverter, the following rule is applied, as shown in equation (2e).

$$R_{inv}C_{inv}\tau \quad (2e)$$

R_{inv} is determined from a delay equation as represented by equation (2f).

$$R_{inv}=\partial d_{inv}/\partial C_{inv} \quad (2f)$$

Applying the same rule to all gates in the library yields equation (2g).

$$(\partial d_{gate}/\partial c_{gate})\,(C_{gate})=(\partial d_{inv}/\partial C_{inv})(C_{inv}) \quad (2g)$$

Throughout the following example, as shown in FIGS. 4A and 4B, it is assumed that the resistance R_{inv} for a P-channel transistor is twice as high as the resistance R_{inv} for an N-channel transistor of the same width. Reference is first made to FIG. 4A which shows a schematic diagram of a real, non-ideal inverter **250**. The inverter **250** includes the P-channel transistor **252** and the N-channel transistor **254**. If it is assumed that the transistors **252** and **254** each have a resistance R_{inv} of 1 unit, then the P-channel transistor **252** has twice the width of the N-channel transistor **254**. Assume further that the input capacitance C_{in} of the inverter **250** is 3 units (wherein the N-channel transistor **254** has an input capacitance of 1 unit and the P-channel transistor **252** has an input capacitance of 2 units). The typical load of the inverter **250** can be derived using the fact that the gain of a non-ideal inverter should be about 3.6. Thus, based on equation (2d), the typical load C_{typ} is shown by equation (2h).

$$C_{typ}=(3.6)C_{in}=(3.6)(3)=10.8 \quad (2h)$$

The typical load value C_{typ} can be used to derive the delay of the inverter **250** wherein $R_{inv}=1$ and $\tau=R_{inv}C_{inv}=(1)(10.8)=10.8$.

Referring now FIG. 4B, the NOR gate **256** includes the P-channel transistors **258** and **260** and the N-channel transistors **262** and **264**. Assuming all the transistors in FIG. 4B have the same size, then the pull-up resistance is 4 units (through both P-channel transistors **258** and **260**) and the pull-down resistance is 1 unit (through either the N-channel transistor **262** or the N-channel transistor **264**). The typical load is determined based on equation (2i).

$$R_{gate}C_{typ}=\tau \quad (2i)$$

For the pull-up resistance wherein $R_{gate}=4$ and $\tau=10.8$, equation (2i) leads to $C_{typ}=2.7$. For the pull-down resistance wherein $R_{gate}1$ and $\tau=10.8$, equation (2i) leads to $C_{typ}=10.8$. It is undesirable to have different answers for C_{typ} for the same gate. Since there can only be one real capacitance in the NOR gate **256**, a number of options are available. For example, the average between the two C_{typ} values (2.7 and

10.8) may be chosen. Alternatively, the most constraining value for C_{typ} may be chosen. In general, such an unbalanced gate as shown in FIG. 4B cannot be tuned to work well for both the rising and falling transitions of a propagating signal. A large discrepancy indicates a poorly dimensioned gate.

As stated above, each gate has more than one delay equation due to multiple gate inputs. Also, different delay equations may correspond to the rising signal transition and to the falling signal transition.

Structural Optimizations

In step **205** (FIG. 4), the invention performs structural optimizations after reading the netlist from step **200**. During this step, the structure of the circuit and the Boolean functions of the gates are changed to reduce the total number of connections, without changing the overall function of the circuit. Structural optimizations can include behavioral optimizations (such as resource sharing), sequential optimizations (such as retiming), algebraic optimizations (such as kernel extraction), and Boolean optimizations (such as redundancy removal). The classes of optimizations above are well known to those skilled in the art.

Mapping

In step **210** (FIG. 4), the circuit is mapped to a library **209** of cells. Thus, the logic functions of the circuit gates are implemented with actual cells from the library **209**. By mapping, circuit delay and other physical attributes can also be determined. In step **210** of the present invention, a conventional mapping process may be used. FIG. **5** shows the mapping process (step **210**) as further including the additional steps of decomposition **300**, cluster generation **305**, matching **310**, and covering **315**. The decomposition step **300** breaks down the large commutative gates into two-input commutative gates.

The cluster generation step **305** selects matches in the logic. Clusters with low chance of success are preferably avoided, such that the total number of clusters to match do not grow very large.

The matching step **310** is the process of finding a library function which can implement the function of the cluster. This step is implemented by use of a conventional matching algorithm such as the algorithm described in K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching", Proceedings of the 24th ACM/IEEE Design Automation Conference, Miami Beach, Fla. (June 1987), pp. 341–347, IEEE Computer Society Press 1987.

The covering step **315** determines which matches are used to actually implement the digital circuit. A timing driven covering method based on dynamic programming may be used in this step. According to this covering method, an implementation of a circuit is chosen from a set of possible implementations based on the arrival time of an implementation. (An arrival time of the data at a gate is computed by taking the maximum arrival time of the fanin gates plus the delay measured from the input pin to the output pin of the gate). An implementation of the circuit with the fastest arrival time may be chosen during the covering step **315**. In contrast to the present invention, a conventional mapping process has a disadvantage of not achieving the accurate calculation of "arrival times," since the arrival times depend on the chosen implementations at multi-fanout points. In the conventional process, the chosen implementations at such multi-fanout points are mutually dependent.

Discussion is now turned to calculation of the arrival time for an implementation of a circuit according to the present invention. Reference is made to FIGS. 6A–6C which show schematic diagrams of portions of a circuit in order to describe arrival time calculation for a gate. In FIG. 6A, the arrival time for a gate **330** (in stage **335**) is determined by the arrival times of its fanin gates (at lines **340**, **345** and **350**) plus the signal delay from input terminals **355** to the output terminals **360** of the gate **330**. The configuration of the fanouts at stage **365** are thus not factored in the arrival time calculation for gate **330**. In FIG. 6B, the arrival time for a gate **370** is determined by the arrival times of its fanin gates **330** and **372** plus the delay between input terminal **375** and output terminal **380** of gate **370**. Similarly, in FIG. 6C, the arrival time for a gate **385** (in stage **387**) is determined by the arrival times of its fanin gates **370** and **390** plus the delay between its input terminal **395** and output terminal **400**.

The process above is repeated for the gate **402** which has an output coupled to an output terminal **404**. Thus, the arrival time at an output terminal **404** of the circuit of FIGS. 6A–6C can be determined based on the arrival time of the gate **402**.

Thus, as shown in FIGS. 6A–6C, in the constant delay approach of the invention, other fanouts of multi-fanouts are irrelevant to the arrival time calculation. The arrival times can thus be computed by traversing the circuit from the inputs to the outputs in a levelized manner. By determining the arrival time for gates on a given path, the total delay of the path in the circuit can be determined.

Level Reduction

Before the cell placement step (step **225** in FIG. 4), it is possible to further revise the circuit than conventionally possible, due to the fact that the delays are held constant. Since the timing constraints are met and the circuit modifications to be made are known, it follows that the effects on delay can be predicted when circuit modifications are made. One specific change or modification which can be made in the mapped circuit is logic level reduction. Reference is made to FIGS. 7A and 7B to illustrate an advantage of using the constant delay model according to the invention. FIG. 7A is a schematic diagram of a gate chain **550** including gates **555**, **560**, **565**, and **570**. Initially, the gate **555** has fanins connected at lines **575** and **580**. The gates **560**, **565** and **570** have fanins connected at lines **585**, **590** and **595**, respectively. A local transformation is then used to reduce the number of levels in the logic in the gate chain circuit **550**. The result of the transformation is shown as gate chain circuit **550'** in FIG. 7B. The number of levels in the logic is reduced by bringing the gate **555** forward. In the constant delay model approach, the effect of this transformation can be easily predicted. Changes in the gate loads do not affect delay, since delay is maintained as constant while gate size will be adjusted (during or after placement) to compensate for the load change. The only change which affects delay (of the gate chain circuit **550**) is the change of the fanin of gate **555**. This delay change can be predicted by simple addition of gate delays provided by the fanins connected at lines **590**, **575**, and **580** (see gate chain circuit **550'** in FIG. 7B).

In order for the transformation shown in FIG. 7B to be valid, it is necessary that gates **555**, **560**, and **565** are fanout free. If the gates **555**, **560**, and **565** are not fanout free, then they are made fanout free through copying logic. For example, assume in the circuit **550** (FIG. 7A) that the gate **555** has an additional fanout **597**. The gate **555** in the circuit **550'** (FIG. 7B) can no longer drive the additional fanout **597** since the gate **555** function may have changed in the circuit **550'**. Thus, in the circuit **550'**, there is provided the copying

logic 555' (which is a duplicate of the gate 555 of circuit 550) for driving the fanout 597.

In contrast, a disadvantage of the conventional design approach is as follows. Under conventional logic design, copying logic will increase the load on the gates whose outputs are connected to lines 575, 580, 585, and 590. In the example of FIG. 7B, the copying logic 555' increases the load on the gates whose outputs are connected to lines 575 and 580. To predict whether or not the transformation improved delay, it is necessary to run a complete static timing analysis with accurate delay models. If the transformation (from circuit 550 to 550') were actually harmful to delay, then the transformation would have to be undone.

Area Analysis and Net Weights Calculations

Discussion is first made on area analysis during which sizes of gates of a mapped circuit are calculated. A gate size is calculated by dividing the gate's actual load by the gate's predetermined typical load C/S. As an example, assume the actual load for a given gate i is expressed as C_i wherein C_i is a capacitance measurement of the gate load. The actual load C_i is approximated by equation (3)

$$C_i=\omega_i+\Sigma C_j*(1/h_{ij}) \quad (3)$$

The parameter ω_i represents the net (wire) load for a given gate i (wherein the net load can be estimated using a conventional net load model such as the above-mentioned fanout-based model) plus any other fixed load such as the load of the primary output of the circuit implementation. The parameter $\Sigma C_j*(1/h_{ij})$ is the input load wherein the subscript j is the fanout gate of the given gate i, and h is the electrical effort between the gates i and j.

Equation (3) can be expressed in matrix notation as shown in equation (4):

$$\bar{c}=\bar{\omega}+(\bar{H})(\bar{c}) \quad (4)$$

wherein $\bar{H}=[1/h_{ij}]$ which is the matrix of the electrical effort. The parameter c is the vector of the gate loads wherein $\bar{c}=[C_i]$ which is the matrix of the gate loads, while $\bar{\omega}$ is the vector of the wire loads ω_i.

According to equations (3) or (4), in order to calculate the size of a given gate i, the size of its fanout gate j must first be calculated, as shown with respect to FIGS. 6A to 6C. The size of the fanout gate j depends on its output capacitive load C_j. This dependency on the fanout gates requires performing the gate load calculation downstream of the data flow. If the digital circuit is a combinational network (see, e.g. circuit 150 in FIG. 2), then gate load calculation initiates at the primary outputs 175 and traverses the circuit in a leveled order toward the primary inputs 170.

If the digital circuit is a sequential network (see, e.g., circuit 180 of FIG. 3), then there may be one or more loops (e.g., loop 182) which result in a cyclic dependency (i.e., there is no "rightmost" gate). Gate load calculation can start anywhere in the cycle, and calculation in the cycle is performed several times until the load capacitance values converge or have sufficiently small differences. However, a condition may exist when the load capacitance values do not converge and increase by progressively larger amounts every cycle calculation. This increase in load capacitance values can be detected if the calculated load values exceed a preset maximum value after a fixed number of cycle calculations. When the calculated load values do not converge, then the particular circuit 180 has an infeasible solution, which indicates that the digital circuit is not expected to work at the set speed because the circuit gain is too small. Changes are required to increase the circuit gain, and these changes will usually lead to an increase in circuit delay.

In the above example, the size S of a gate i is determined by dividing the actual load C_i by the predetermined typical load C/S of the gate i. The size S is a scale factor which is applied to all transistor channel widths of a gate in order to determine the area of the "sized gate". The size S is also a scale factor which is used to scale the gate's output load driving capability and its input pin loads.

The area of the sized gate is determined by equation (5).

$$\text{area of sized gate}=S*(\text{area of gate}) \quad (5)$$

The area of the mapped digital circuit can be estimated based on the sum of the total areas of the sized gates plus the net area (which is estimated from the total length of all nets in the circuit).

During the mapping step, possible circuit implementations may be evaluated based on net weight calculations. Thus, the following discussion now turns to the calculation of "net weights." The net weight represents the sensitivity of the total area of a digital circuit with respect to the load of a particular net. As an example, the net weight of a given gate, which is immediately coupled to the primary inputs of a digital circuit, is equal to its area per unit load. Using equation (6), the net weight of the other gates in the digital circuit are then calculated in a leveled order towards the primary outputs of the digital circuit.

$$w_i=\partial A/\partial C_i=\partial a_i/C_i+\Sigma w_j*(1/h_{ij}) \quad (6)$$

In equation (6), the subscript j represents the fanin gate of a given gate i. The parameter w_i is the net weight of the gate i, while A is the total area of the digital circuit, a_i is the area of the gate i, and C_i is capacitive load of the gate i. As stated above, h_{ij} is the electrical effort between a gate i and j.

Equation (6) can be expressed in matrix notation as shown in equation (7):

$$\bar{w}=\bar{\alpha}+(\bar{H})(\bar{w}) \quad (7)$$

wherein $\bar{H}=[1/h_{ij}]$, $\bar{\alpha}=[\partial a_i/\partial C_i]$, and $\bar{w}=[w_i]$.

The net weight value from equation (6) can be also used to later calculate the cell area of the cells after the placement step (step 225 in FIG. 4). This cell area is expressed by Equation (8):

$$A=\Sigma(w_i)((\omega_i) \quad (8)$$

wherein the parameter ω_i represents the wire load for a given gate i.

During mapping, the calculated net weight value can be used to determine which matches are used to implement the digital circuit, since the calculated net weight value makes it possible to take into consideration the circuit area. As shown in equation (6), the net weight for a given gate i only depends on the logic of the fanin cone (which was already mapped), wherein the net weight represents the sensitivity of the total area of the circuit with respect to the load on a cell. As further shown in equation (8), the area of a cell increases if its net weight increases. Thus, for example, assume that for a desired circuit function, a first circuit implementation has a higher net weight than the net weight of a second circuit implementation. The first circuit implementation will therefore have a larger area for any possible load than the second implementation. Therefore, disregarding delay, the second circuit implementation is more preferable than the

first circuit implementation, even though the area values of both circuit implementations may have not been determined at this stage of the design process.

As shown by the above example, the net weight calculations permit circuit implementation decisions to be made, partly based on the gate area. The net weight calculations permit the area of one cell to be compared relatively to the area of another cell wherein both cells have the same function in the desired digital network. An advantage of the net weight calculations is that when comparing the area of cells, the actual area values of the cells are not required to be known.

Buffering

Prior to cell placement, certain matches may be changed by insertion of buffers in the digital circuit, as shown in step 215 in FIG. 4. Under this step, adding a buffer will increase delay, but also save on area since the source gate can be smaller in size due to a smaller load. Large loads are preferably driven by buffers, particularly when the signal timing is not critical. Buffering allows the source cell to be sized down, thereby resulting in a reduction in circuit area.

The buffering step of 215 (FIG. 4) is discussed in further detail with reference to FIG. 8. In step 650, locations in the circuit are determined where a buffer can be added so that buffer insertion will still permit timing constraints to be met. This determination is made by subtracting the delay of the buffer from the "local slack", to give the value of the predicted slack after buffer insertion. Slack is zero or positive if the timing constraints are met. In addition, all slacks in the circuit can be summarized by the "network slack" which is the single "worst" slack number. If the network slack is non-negative, then timing closure is achieved. If the predicted slack calculated in step 650 is larger than the network slack, then it is possible to insert a buffer without increasing the circuit delay.

In step 655, it is determined whether the added area due to buffer insertion does not exceed the area saved by sizing down the source gate. The added area (by inserting the buffer) is equal to the area of the buffer multiplied by the buffer size, wherein the buffer size is determined by the buffer load C divided by the typical load C/S on the buffer. The area saved by sizing down the source gate is determined by first calculating the change in net load due to the buffer insertion. This net load change is due to the following: (1) some sinks (which sink currents) are removed, (2) the input load of the buffer is added, and (3) the number of fanouts of the gate may change.

Using the net weight equation (6), the impact of buffer insertion on the work area can be estimated (step 660). In step 665, the buffer is inserted if the impact on the circuit area is positive (i.e., the circuit area is reduced). After the buffer has been inserted, then in step 670 the capacitance values need to be updated in the fanin cone of the buffer, while the net weights need to be updated in the fanout cone of the buffer. This updating step serves to keep the net weights accurate, while limiting the net weight re-calculation to those portions of the circuit affected by buffer insertion.

FIG. 8A shows a gate 680 which has buffers 685 and 690 inserted in the gate 680 output load. The buffers 685 and 690 permit the gate 680 to be sized down. The buffers 685 and 690 are inserted if the area impact on circuit 692 is positive (i.e., the area of circuit 692 is reduced due to buffering).

Buffer insertion is a method for optimizing the circuit area. As a result of buffer insertion, the timing of the circuit becomes slower, but the circuit timing will still meet the timing constraints. Buffer insertion causes the net delay to increase due to the added delay from the buffer. For those portions of the circuit where timing constraints have already been met, adding buffers can still result in meeting the previously determined timing constraints.

While it is preferred to stay within the timing constraints, it is within the scope of the present invention to go beyond the timing constraints and to correct the circuit later in the design process so that the timing constraints are eventually met. Additionally, where it is determined that saving area is of more significance than in meeting timing constraints, buffers can be inserted to save area, even though the previously determined timing constraints are not met. Many paths in a circuit do not have the critical timing requirements, and thus, buffers can be inserted in these paths to save area.

Compressing or Stretching of Gate Delays

Prior to cell placement, the delays of the individual gates may be stretched or compressed to meet the delay constraints, as shown in step 220 of FIG. 4. As shown in FIG. 9A, by compressing (decreasing) the delay of a given gate, the gate gain decreases. Gates which are on long paths not meeting the delay constraints are compressed (in delay) until the long paths meet the delay constraints. The delay of the gates (or gate) may be decreased as long as the minimum required gain requirements are met.

By stretching (increasing) the delay of a given gate, the gate gain increases (see FIG. 9A). Gates on short paths which easily meet the delay constraints are stretched (in delay), since gates with stretched delays require less area for the same load. The delay of the gates (or gate) in a path are stretched to the extent that timing constraints for the digital circuit are still met.

The step of stretching and compressing of a gate delay is not related to the adjustment of gate sizes. As stated above, to stretch or compress the gate delay, the gate gain is appropriately adjusted. The stretching and compressing step is discussed in further detail with reference to FIG. 9 which shows a portion of a digital circuit 750. As an example, the gate 752 has a requirement of driving an output capacitance load (C_{out}) of about 2.0 pico-farads, while having an input capacitance load (C_{in}) of about 0.1 pico-farads. Thus the gain requirement of the gate 752 is $C_{out}/C_{in}=2.0/0.1$. The gate 754 is assumed to have an output capacitance load of about 2.0 pico-farads and an input capacitance load of about 0.2 pico-farads. Thus, the gain requirement of the gate 754 is 2.0/0.2.

It is also assumed that the gate 752 has a delay, D=1. It is assumed further that the gate 752 is on a "Path 1" which has an arrival time requirement of about 1 at its beginning point 756 and an arrival time requirement of about 4 at its end point 758. Thus, the delay constraint for Path 1 is equal to about 3, as shown by equation (9):

$$\text{constraint}=\text{arr. time (point 756)}-\text{arr. time (point 758)}=4-1=3 \quad (9)$$

Since the delay D of gate 752 equals 1 and the delay constraint of Path 1 equals 3, the slack for Path 1 equals 2, as shown by equation (10):

$$\text{slack}=\text{delay constraint}-\text{gate(s) delay(s)}=3-1=2 \quad (10)$$

It is further assumed that the gate 754 has a delay, D=5. It is assumed further that the gate 754 is on a "Path 2" which has an arrival time requirement of about 1 at its beginning

point 760 and an arrival time requirement of about 4 at its end point 762. Thus, the delay constraint for Path 2 is equal to about 3, as shown by equation (11):

$$\text{constraint} = \text{arr. time (point 760)} - \text{arr. time (point 762)} = 4 - 1 = 3 \quad (11)$$

Since the delay of gate 754 (D=5) is greater than the Path 2 delay constraint of 3, the delay of gate 754 does not meet the timing constraints.

The invention operates on a path-by-path basis whereby the most critical path in a digital circuit 750 is evaluated first. Thus, Path 1 is evaluated first if it is the most critical path. Path 2 is evaluated first if it is the most critical path. Assuming Path 2 is the most critical path, then the delay of the gate 754 is compressed to meet the Path 2 delay constraint of 3, while observing the effects on the gain of the gate 754. As stated above, the gain requirement of the gate 754 is C_{out}/C_{in}=2.0/0.2. However, as shown in FIG. 9A, the gain of any particularly sized gate decreases as the gate delay D is decreased. Thus, compression of the gate 754 delay may be limited by its gain requirements.

After the gate 754 has been adjusted to meet the Path 2 timing constraints, it becomes "locked," whereby the gate 754 delay will not be adjusted further for the remainder of the compression and stretching step. It is further noted that if a path contains more than one gate, then, preferably, their delays are compressed (or stretched) in equal amounts to meet the path timing constraints. For example, assume that four (4) gates are in a given path and two (2) nano-seconds of delay can be added to the given path. Then according to a preferred embodiment of the invention, 0.5 nano-seconds of delay can be added to each of the four (4) gates to equally distribute the two nano-seconds of delay which is to be added in the given path.

The invention then proceeds to evaluate the next critical path. Assume that Path 1 is the next most critical path. In FIG. 9, the delay of gate 752 (D=1) meets the Path 1 delay constraint of 3, and thus the gate 752 delay can be stretched based on a slack of 2. The gate 752 is then locked after its delay is stretched. If a plurality of gates are on Path 1, then preferably, their delays are stretched in equal amounts.

The invention will then evaluate the next critical path, whereby gate delays may be stretched or compressed. When reducing slack values during the gate delay stretching step, it is preferable to maintain a small amount of slack for a path, in order to compensate for downstream delay effects, such as wire delay. Additionally, in some instances, library rules may limit the amount of slack value reduction.

For the purpose of stretching and compressing, registers in the circuit are preferably considered as part of a path from which they originate, but not part of the path from which they terminate. The stretching and compressing steps are further shown in the flowchart of FIG. 9B. In step 770, the most critical path on the circuit is evaluated, and in step 775, the delay of a gate (or gates) in the path are stretched or compressed. In step 780, the next critical path is evaluated, and in step 782, the delay of a gate (or gates) in the path are stretched or compressed. In step 784, if all paths have not been evaluated, then the invention returns to step 780 to evaluate the next critical path. If in step 784 all paths have been evaluated, then the invention proceeds to step 785 wherein placing of cells may be performed.

Cell Placement

In step 225 (FIG. 4), the placement of cells is performed. FIGS. 10A and 10B show how a circuit design is transformed from a logical hierarchy 830 to a physical hierarchy 832 during the cell placement step 225 (FIG. 4). In the physical hierarchy 832, the intermediate logic levels in the logical hierarchy 830 are associated in or grouped in buckets 834 wherein each bucket 834 holds, for example, about one-hundred (100) cells 836. The buckets 834 are arranged in an array as shown in FIG. 10C. If the designer chooses to keep a group of cells 836 together, then the group of cells 836 are grouped in the same buckets 834 or in neighboring buckets. Preferably, a bucket 834 is sized small enough such that cell placement within a bucket 834 has an insignificant effect on timing. In other words, the size of a bucket 834 is such that the wire delay in a bucket 834 can be ignored. However, the size of a bucket 836 should be large enough to accommodate remapping and resizing of the cells 836 contained in the bucket. The number of cells which can be placed within a bucket can range, preferably, from about 20 cells to about 200 cells.

Pre-routes and pre-places (for driving the placer and global router) are driven into the bucket 834 structure. It is further noted, however, that the present invention may be practiced or incorporated with conventional placement methods and systems.

According to the present invention, timing closure is maintained after placement occurs of cells 836. To maintain timing closure, the size of a particular gate may be adjusted during or after placement. This adjustment compensates for the fact that placement algorithm may assign different net lengths to different nets, and that these different net lengths are difficult to predict prior to the placement step. Thus, by the practice of the present invention, the gate sizes are adjusted in order to maintain the delay values which were determined prior to placement.

In contrast, conventional placement methods optimize the length of wires to compensate for changes in delays, due to net length changes caused by the placement algorithm. Thus, conventional placement methods involve costly and lengthy delay analysis avoided by the present invention.

As stated above, equation (8) determines the cell area of the placed cells:

$$A = \Sigma(w_i)(\omega_i) \quad (8)$$

As also stated above, the parameter ω_i represents the wire load for a given gate (e.g., gate i). Based on equation (8), the area for each gate is determined based on the product, $(w_i)(\omega_i)$. Thus, a given gate i can be sized appropriately, and avoidance is possible of the costly delay analysis required in conventional methods when the actual net lengths differ from the estimated net lengths prior to placement.

Routing of the Digital Circuit

Finally, in step 230 (FIG. 4), routing of the placed circuit is performed to complete the chip layout 235. The routing step places wires which were previously determined during or prior to the placement step. In step 230, gate delay and gate size are fixed. The router thus controls the wire loads to maintain the delay as essentially constant and thus achieve timing closure. Gate delay and gate size have been previously fixed. The router thus controls the distance 900 (FIG. 11) between, for example, wires 905 and 910. The capacitance between the two wires 905 and 910 decreases as the distance 900 between wires increases. A conventional router can be used to control the distance 900.

This specification shows the components (e.g., buffers, cells, nets) which exist when the integrated circuit layout is produced. However, it is realized by those skilled in the art

that when performing the steps in accordance with the present invention prior to producing the layout, the shown components are data representation stored in the computer and not the actual devices.

Thus, while the present invention has been described herein with reference to particular embodiments thereof, a latitude of modification, various changes and substitution are intended in the foregoing disclosure, and it will be appreciated that in some instances some features of the invention will be employed without a corresponding use of other features without departing from the scope of the invention as set forth.

What is claimed is:

1. An automated method for designing an initial integrated circuit layout of a digital circuit with a computer, based upon an electronic circuit description and by using a cell library containing cells, comprising the steps of:

(a) selecting a plurality of cells from the cell library that are intended to be coupled to each other with a plurality of wires and that can be used to implement the digital circuit based on the electronic circuit description input to the computer to obtain a selected plurality of cells, at least some of the selected plurality of cells having an initial intended delay associated therewith for ensuring that predetermined timing constraints are met;

(b) determining a placement of the selected plurality of cells and the wires coupled thereto using a sequence of instructions from the computer program; and

(c) determining the area of the some cells, the area of each some cell being determined using the lengths of the wires coupled to each of said some cells such that the initial intended delay of each some cell is realized, the length of each wire being determined by the placement of the cells coupled to that wire.

2. The automated method of claim 1 further comprising:

routing the digital circuit to generate the integrated circuit layout.

3. The automated method of claim 2 wherein,

prior to the step of routing, the area and placement of each of the selected plurality of cells and the lengths of the wires is finalized.

4. The method of claim 1 wherein the plurality of cells are coupled to each other based on the electronic circuit description input to the computer.

5. The automated method of claim 1 wherein each of the plurality of wires has an associated capacitive load value; and

wherein each wire is associated with a net weight value that represents the sensitivity of the total area of the digital circuit of step (a) with respect to the associated capacitive load value of the wire.

6. The automated method of claim 5 wherein the net weight w_i of the wire coupled to a selected cell is:

$$w_i = \delta A / \delta C_i$$

where A is the total area of the digital circuit of step (a) and C_i is the associated capacitive load of the wire coupled to the selected cell.

7. The automated method of claim 6 wherein the determined areas of said some selected cells are chosen based upon the product of the net weight w_i and the net load C_i in order to meet said predetermined timing constraints.

8. The automated method of claim 1 wherein all of the selected cells have an initial intended delay associated therewith.

9. The automated method of claim 8 wherein the step of determining determines the area of all of the selected cells.

10. The automated method of claim 1 further comprising step of inserting a buffer in one of the wires after the step of determining the placement and before the step of determining the area.

11. The automated method of claim 10 wherein the step of inserting the buffer comprises:

determining the wire into which a the buffer can be inserted while still meeting the predetermined timing constraints; and

inserting the buffer into the wire if area will be saved by the insertion.

12. The automated method of claim 1 further comprising:

stretching the initial intended delay of a selected cell to decrease the area of the selected cell after the step of determining the placement and before the step of determining the area.

13. The automated method of claim 1 further comprising:

stretching the initial intended delays of a plurality of selected cells coupled to each other to decrease the area of each of the coupled cells after the step of determining the placement and before the step of determining the area.

14. The automated method of claim 1 further comprising:

compressing the initial intended delay of a selected cell after the step of determining the placement and before the step of determining the area.

15. The automated method of claim 14 wherein the compressing step is limited by a gain requirement of the selected cell.

16. The automated method of claim 1 further comprising:

compressing the initial intended delays of a plurality of the selected cells after the step of determining the placement and before the step of determining the area.

17. The automated method of claim 1 wherein a group of said some cells are assigned in buckets and operated upon.

18. The automated method of claim 17 wherein the group of said some cells ranges from 20 to 200 cells.

19. An integrated circuit layout produced in accordance with the automated method of claim 1.

20. The method according to claim 1 wherein the cell library for each of the some cells selected does not include the area of each of the some cells.

21. An automated method for designing an integrated circuit layout using a computer, based upon an electronic circuit description and based upon cells which are selected from a cell library, comprising the steps of:

(a) placing each of the cells in the integrated circuit layout and wires associated therewith so that the cells can be coupled together by wires to form a circuit path having an associated predetermined timing constraint wherein the cells are coupled together by wires based upon the electronic circuit description input to the computer, the electronic circuit description including an initial intended delay associated with each cell; and

(b) determining an area of at least some of the cells to satisfy the associated predetermined timing constraint of the circuit path, the area being determined as a function of the ratio of the input capacitance and output capacitance of the wires coupled to each of said some cells such that the initial intended delay of each cell is realized.

22. The automated method of claim 21 wherein each of the wires has an associated capacitive load value; and

wherein each wire is associated with a net weight value that represents the sensitivity of the total area of the

integrated circuit layout with respect to the associated capacitive load value of the wire.

23. The automated method of claim 22 wherein the net weight w_i of the wire coupled to a cell is:

$$w_i = \delta A / \delta C_i$$

where A is the total area of the integrated circuit layout and C_i is the associated capacitive load of the wire coupled to the cell.

24. The automated method of claim 23 wherein the areas of at least some of the cells are chosen based upon the product of the net weight w_i and the net load C_i in order to meet said predetermined timing constraints.

25. The automated method of claim 24 wherein the length of each wire is determined by the placement of the cells coupled to that wire.

26. The automated method of claim 21 wherein the length of each wire is determined by the placement of the cells coupled to that wire.

27. The automated method of claim 21 further comprising:

inserting a buffer in one of the wires after the step of placing and before the step of determining the area.

28. The automated method of claim 21 further comprising:

stretching the associated relative delay value of a selected cell after the step of placing and before the step of determining the area.

29. The automated method of claim 21 further comprising:

compressing the associated relative delay value of a selected cell after the step of placing and before the step of determining the area.

30. The automated method of claim 29 wherein the compressing step is limited by a gain requirement of the selected cell.

31. The automated method of claim 21 wherein a group of said some cells are assigned in buckets and operated upon in order to determined the initial intended area of each of the group of said some cells.

32. The automated method of claim 31 wherein the group of said some cells ranges from 20 to 200 cells.

33. The automated method of claim 21 wherein the step of determining determines the area of all of the selected cells.

34. The method according to claim 21 wherein the electronic circuit description used to placing each of the cells in the integrated circuit layout does not include the area of the cells.

35. An automated method for designing an integrated circuit layout of a circuit of at least four cells coupled to each other with a plurality of wires by using a computer and based upon an electronic circuit description containing information on the digital circuit, comprising the steps of:

(a) selecting a plurality of cells which can be used to implement the digital circuit based upon the electronic circuit description, each of the plurality of cells having an associated load and an initial intended delay associated therewith for ensuring that predetermined timing constraints are met, the selected plurality of cells comprising at least a first cell having a first load, a second cell having a second load, a third cell having a third load, and a fourth cell having a fourth load;

(b) determining initial placement locations for each of the selected plurality of cells, including the first cell, the second cell, the third cell, and the fourth cell;

(c) setting the size of each of the selected plurality of cells and the loads of each cell so that the predetermined timing constraints are met, the size of each cell being determined using the lengths of the wires coupled to each of said cells such that the initial intended delay of each cell is realized, the length of each wire being determined by the placement of the cells coupled to that wire.

36. The automated method of claim 35 wherein each of the load has an associated capacitive load value; and

wherein each load is associated with a net weight value that represents the sensitivity of the total area of the integrated circuit layout with respect to the associated capacitive load value of the load.

37. The automated method of claim 36 wherein the net weight w_i of the load coupled to a cell is:

$$w_i = \delta A / \delta C_i$$

where A is the total area of the integrated circuit layout and C_i is the associated capacitive load of the load coupled to the cell.

38. The automated method of claim 37 wherein the areas of at least some of the cells are chosen based upon the product of the net weight w_i and the net load C_i in order to meet said predetermined timing constraints.

39. The method according to claim 35 wherein the plurality of cells are selected without having a size associated therewith.

40. A method according to claim 35 in which each of the first, second, third and fourth cells implement at least one logic operation.

41. A method according to claim 35 which none of the first, second, third and fourth cells are buffers.

42. An integrated circuit layout produced in accordance with the automated method of claim 35.

43. An automated method for determining an integrated circuit layout of at least four cells by using a computer and based upon an electronic circuit description containing information on the digital circuit, the automated method comprising the steps of:

(a) selecting a plurality of cells which can be used to implement the digital circuit using the electronic circuit description, the plurality of cells comprising a first cell having a first load and a first initial intended delay, a second cell connected to the first cell and having a second load and a second initial intended delay associated therewith, a third cell connected to the second cell and having a third load and a third initial intended delay associated therewith, and a fourth cell connected to the third cell and having a fourth load and a fourth initial intended delay associated therewith, the digital circuit having predetermined timing constraints;

(b) determining the placement locations for each of the selected plurality of cells including the first cell, the second cell, the third cell, and the fourth cell and wires associated therewith;

(c) selecting the size of the first cell based on the first load of the first cell so that the first initial intended delay is realized, the size of the first cell being determined using the lengths of the wires coupled to the first cell, the length of each wire being determined by the placement of the cells coupled to that wire;

(d) selecting the size of the second cell based on the second load of the second cell so that the second initial intended delay is realized, the size of the second cell being determined using the lengths of the wires coupled

to the second cell, the length of each wire being determined by the placement of the cells coupled to that wire;

(e) selecting the size of the third cell based on the third load so that the third initial intended delay is realized, the size of the third cell being determined using the lengths of the wires coupled to the third cell, the length of each wire being determined by the placement of the cells coupled to that wire; and

(f) selecting the size of the fourth cell based on the fourth load so that the fourth initial intended delay is realized, the size of the fourth cell being determined using the lengths of the wires coupled to the fourth cell, the length of each wire being determined by the placement of the cells coupled to that wire, wherein the selection of the area of the first, second, third and fourth cells ensures that the predetermined timing constraints associated therewith are met.

44. An integrated circuit layout produced in accordance with the automated method of claim 43.

45. A method according to claim 43 further including the step of routing the digital circuit.

46. A method according to claim 43 in which each of the first, second, third and fourth cells implement at least one logic operation.

47. A method according to claim 43 in which none of the first, second, third and fourth cells are buffers.

48. The method according to claim 43 wherein the plurality of cells are selected without having a size associated therewith.

49. An automated method of modeling the delay of the cells of an integrated circuit comprising the steps of:

associating an initial gain value with each cell that has been initially selected for inclusion in the integrated circuit;

computing the initial intended delay value of each cell based on the initial intended gain value.

50. The automated method of claim 49, wherein the initial intended gain value is determined such that the variation in the variable component of the initial intended delay is the same for all the cells in the circuit is minimized.

51. The automated method of claim 49, wherein the associated gain value of one or more cells are reduced to compress the associated relative delay value of said cells to assist in satisfying the predetermined timing constraints.

52. The automated method of claim 49, wherein the associated gain value of one or more cells are increased to stretch the associated relative delay value of said cells to assist in reducing the area of said cells.

53. The automated method of claim 52, further including the step of computing the input capacitance of each cell, the input capacitance being computed as the capacitive output load of said cell divided by the gain.

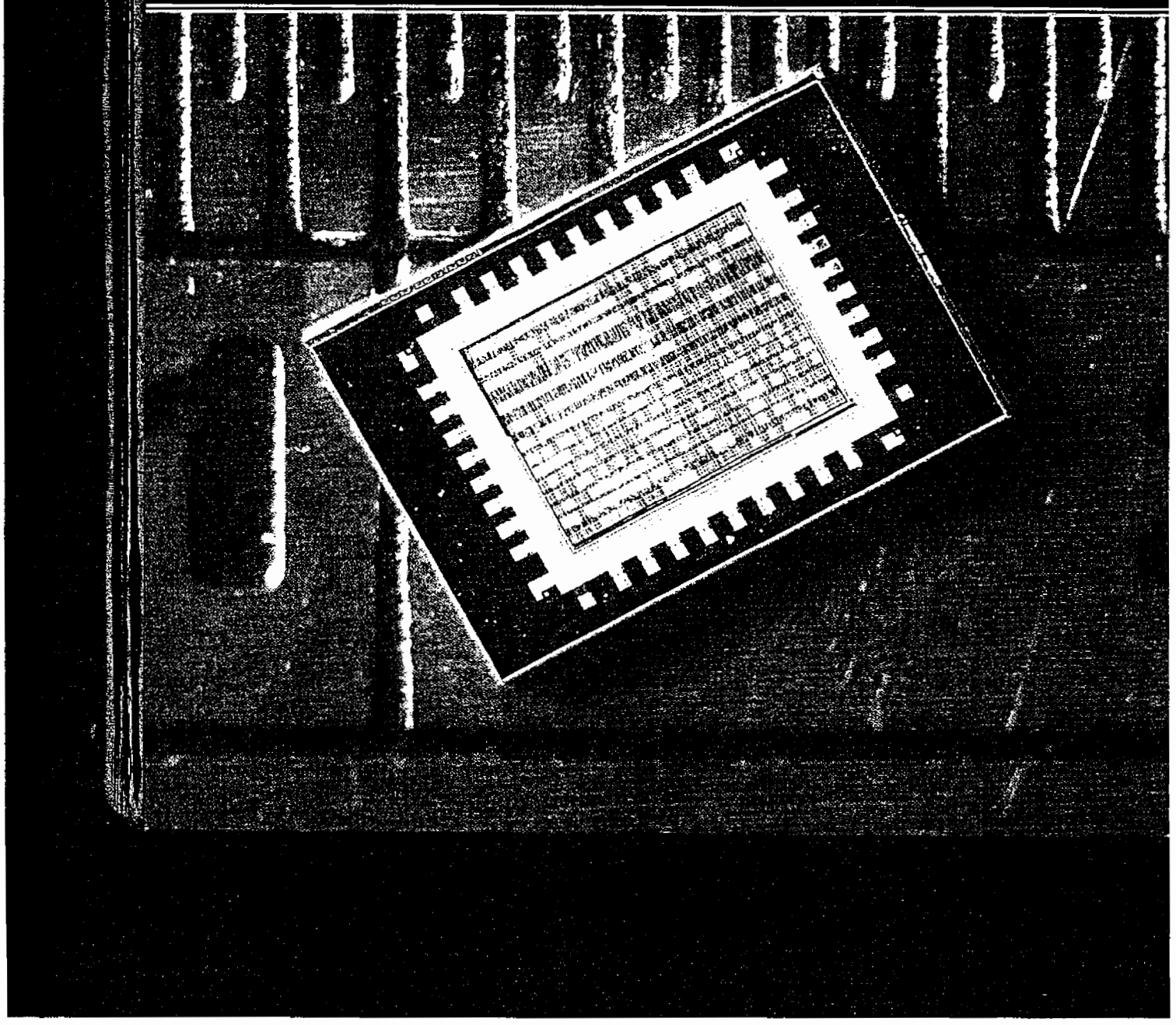
54. The automated method of claim 53, further including the step of computing the area of each cell, the area of each cell being computed as the capacitive output load of said cell times the net weight of each cell.

* * * * *

TAB 17

Application-Specific Integrated Circuits

Michael John Sebastian Smith



Application-Specific Integrated Circuits

Michael John Sebastian Smith
University of Hawaii
Compass Design Automation

ADDISON-WESLEY

An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Harlow, England • Menlo Park, California • Berkeley, California
Don Mills, Ontario • Sydney • Bonn • Amsterdam • Tokyo • Mexico City

This book is in the Addison-Wesley VLSI Systems Series
Lynn Conway and Charles Seitz, *Consulting Editors*

Sponsoring Editor	Peter Gordon
Associate Editor	Helen Goldstein
Senior Production Supervisor	Juliet Silveri
Copyeditor/Proofreader	Cynthia Benn
Cover Design Supervisor	Simone Payment
Marketing Manager	Tracy Russ
Manufacturing Manager	Roy Logan

Library of Congress Cataloging-in-Publication Data

Smith, Michael J. S. (Michael John Sebastian)
Application-specific integrated circuits / Michael J.S. Smith.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-50022-1
1. Application-specific integrated circuits. I. Title.
TK7874.6.S63 1997
621.39'5--dc20 93-32538
CIP

Access the latest information about Addison-Wesley books from our World Wide Web page:
`http://www.awl.com/cseng`

1 2 3 4 5 6 7 8 9 10-CRW-00999897

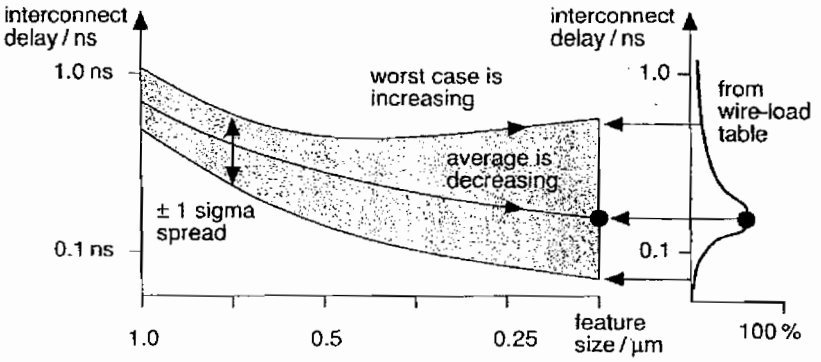


FIGURE 16.5 Worst-case interconnect delay. As we scale circuits, but avoid scaling the chip size, the worst-case interconnect delay increases.

16.1.3 Floorplanning Tools

Figure 16.6(a) shows an initial **random floorplan** generated by a floorplanning tool. Two of the blocks, A and C in this example, are standard-cell areas (the chip shown in Figure 16.1 is one large standard-cell area). These are **flexible blocks** (or **variable blocks**) because, although their total area is fixed, their shape (aspect ratio) and connector locations may be adjusted during the placement step. The dimensions and connector locations of the other **fixed blocks** (perhaps RAM, ROM, compiled cells, or megacells) can only be modified when they are created. We may force logic cells to be in selected flexible blocks by **seeding**. We choose **seed cells** by name. For example, `ram_control*` would select all logic cells whose names started with `ram_control` to be placed in one flexible block. The special symbol, usually '*', is a **wildcard symbol**. Seeding may be hard or soft. A **hard seed** is fixed and not allowed to move during the remaining floorplanning and placement steps. A **soft seed** is an initial suggestion only and can be altered if necessary by the floorplanner. We may also use **seed connectors** within flexible blocks—forcing certain nets to appear in a specified order, or location at the boundary of a flexible block.

The floorplanner can complete an estimated placement to determine the positions of connectors at the boundaries of the flexible blocks. Figure 16.6(b) illustrates a **rat's nest** display of the connections between blocks. Connections are shown as **bundles** between the centers of blocks or as **flight lines** between connectors. Figure 16.6(c) and (d) show how we can move the blocks in a floorplanning tool to minimize routing **congestion**.

We need to control the **aspect ratio** of our floorplan because we have to fit our chip into the **die cavity** (a fixed-size hole, usually square) inside a package. Figure 16.7(a)–(c) show how we can rearrange our chip to achieve a square aspect ratio. Figure 16.7(c) also shows a **congestion map**, another form of **routability** display. There is no standard measure of routability. Generally the **interconnect channels**, (or wiring channels—I shall call them channels from now on) have a cer-

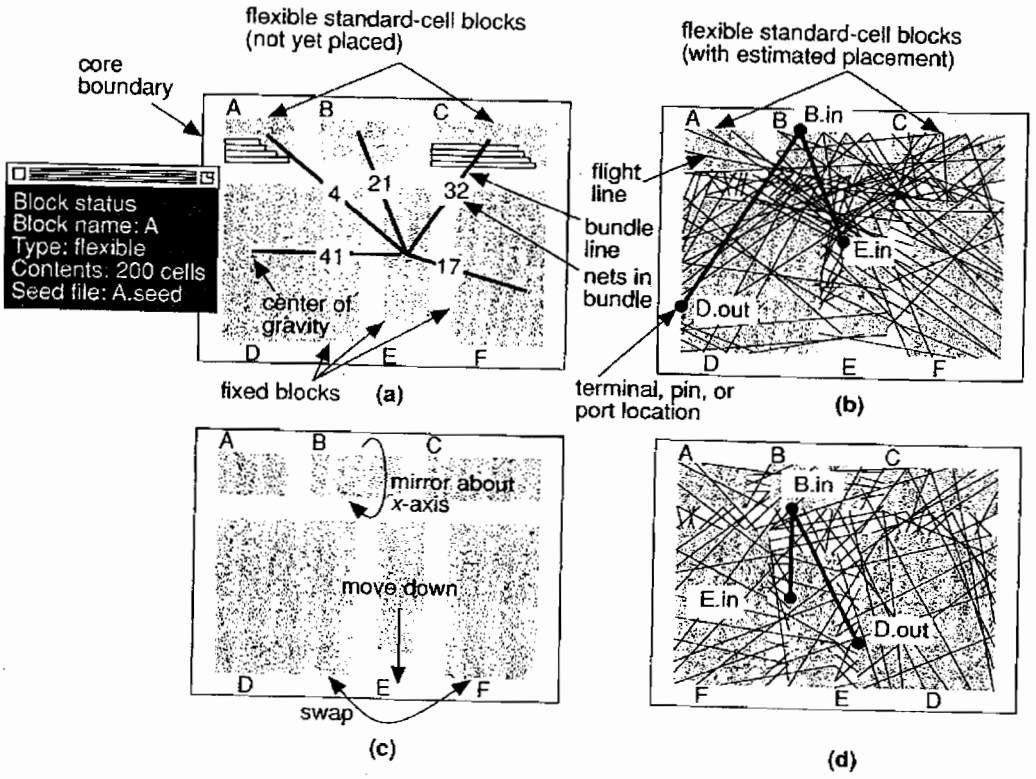


FIGURE 16.6 Floorplanning a cell-based ASIC. (a) Initial floorplan generated by the floorplanning tool. Two of the blocks are flexible (A and C) and contain rows of standard cells (unplaced). A pop-up window shows the status of block A. (b) An estimated placement for flexible blocks A and C. The connector positions are known and a rat's nest display shows the heavy congestion below block B. (c) Moving blocks to improve the floorplan. (d) The updated display shows the reduced congestion after the changes.

tain **channel capacity**; that is, they can handle only a fixed number of interconnects. One measure of congestion is the difference between the number of interconnects that we actually need, called the **channel density**, and the channel capacity. Another measure, shown in Figure 16.7(c), uses the ratio of channel density to the channel capacity. With practice, we can create a good initial placement by floorplanning and a pictorial display. This is one area where the human ability to recognize patterns and spatial relations is currently superior to a computer program's ability.

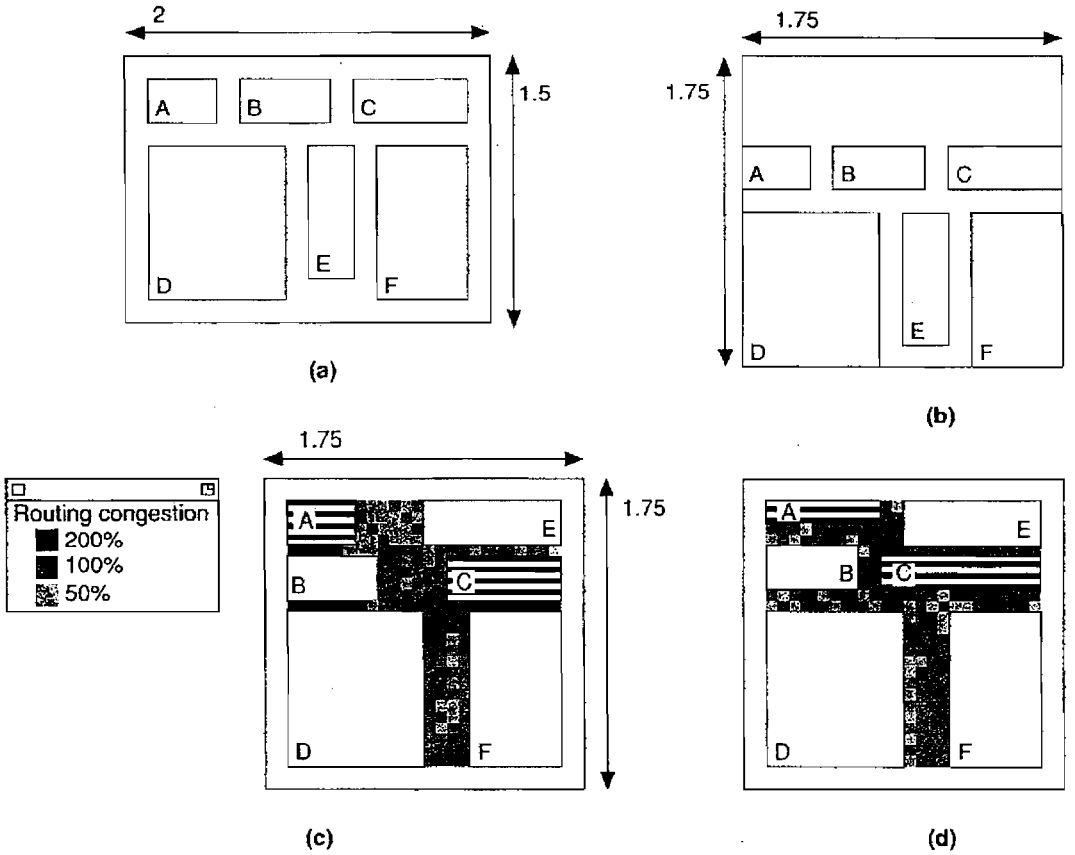


FIGURE 16.7 Congestion analysis. (a) The initial floorplan with a 2:1.5 die aspect ratio. (b) Altering the floorplan to give a 1:1 chip aspect ratio. (c) A trial floorplan with a congestion map. Blocks A and C have been placed so that we know the terminal positions in the channels. Shading indicates the ratio of channel density to the channel capacity. Dark areas show regions that cannot be routed because the channel congestion exceeds the estimated capacity. (d) Resizing flexible blocks A and C alleviates congestion.

16.1.4 Channel Definition

During the floorplanning step we assign the areas between blocks that are to be used for interconnect. This process is known as **channel definition** or **channel allocation**. Figure 16.8 shows a T-shaped junction between two rectangular channels

TAB 18

US006618846B2

(12) **United States Patent**
Cheng

(10) **Patent No.: US 6,618,846 B2**
(45) **Date of Patent: Sep. 9, 2003**

(54) **ESTIMATING CAPACITANCE EFFECTS IN INTEGRATED CIRCUITS USING CONGESTION ESTIMATIONS**

(75) Inventor: **Chih-liang Cheng**, Fremont, CA (US)

(73) Assignee: **Synopsys, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/944,885**

(22) Filed: **Aug. 31, 2001**

(65) **Prior Publication Data**

US 2003/0051217 A1 Mar. 13, 2003

(51) **Int. Cl.**[7] **G06F 17/50**

(52) **U.S. Cl.** **716/5**; 716/9; 716/8; 716/11; 716/13

(58) **Field of Search** 716/1–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,587,923	A		12/1996	Wang
5,798,936	A		8/1998	Cheng
5,831,870	A	*	11/1998	Folta et al. 716/5
5,847,965	A		12/1998	Cheng
6,175,947	B1	*	1/2001	Ponnapalli et al. 716/5
6,327,693	B1	*	12/2001	Cheng et al. 716/12
6,415,422	B1	*	7/2002	Mehrotra et al. 716/5

OTHER PUBLICATIONS

Maogang Wang, Xiaojian Yang, Majid Sarrafzadeh, "Congestion Minimization During Placement", *IEEE Transactions on Automatic Control, vol. XX, No. Y. Month 1999*, pps. 100–109.

Joseph L. Ganley, "Computing Optimal Rectilinear Steiner Trees: A Survey and Experimental Evaluation", *Cadence Design Systems, Inc.* 2615 John Milton Drive, Oak Hill, Virginia 20171, Preprint submitted to Elsevier Science, pps. 1–13, Feb. 17, 1998.

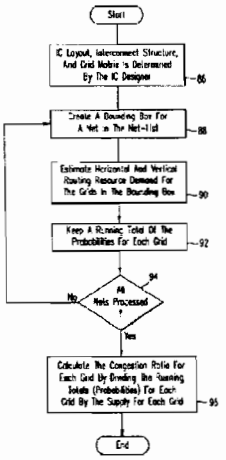
(List continued on next page.)

Primary Examiner—Matthew Smith
Assistant Examiner—Magid Dimyan
(74) *Attorney, Agent, or Firm*—Silicon Valley Patent Group LLP

(57) **ABSTRACT**

A method estimates the capacitance effects of an interconnect prior to routing of an integrated circuit (IC) design, as follows. The design is divided into areas. Capacitance effects for each area are estimated based on the congestion ratios within the area. The congestion ratios for each area are derived from estimations of the demand for routing resources in each area for each net in the net-list included in the IC design. Coupling vectors are derived for each area from the congestion ratios. Capacitance effects for each area are then estimated by looking up a database using the coupling vectors. The resulting per-area capacitance effects are then used to estimate capacitance in an interconnect traversing the area. The total capacitance effects due to an interconnect traversing multiple areas is determined by applying the per-area capacitance effects for the areas to the dimensions of portions of the interconnect traversing each of the areas.

24 Claims, 13 Drawing Sheets



OTHER PUBLICATIONS

"Fast Congestion Prediction by Rent's Rule", 5 pps.

Moagang Wang, Xiaojian Yang, Majid Sarrafzadeh, "Multi–Center Congestion Estimation and Minimization During Placement", *Department of Electrical and Computer Engineering, Northwestern University*, Evanston, IL 60208, 6 pps.

Ion I. Măndoiu, Vijay V. Vazirani, Joseph L. Ganley, "A New Heuristic for Rectilinear Steiner Trees", *Supported by NSF Grant CCR 9627308*, pps. 1–6.

Gabriel Robins and Alexander Zelikovsky "Improved Steiner Tree Approximation in Graphs", *Department of Computer Science, University of Virginia and Georgia State University, Supported in part by a Packard Foundation Fellowship, and by a GSU Resarch Initiation Grant*, pps. 1–11.

Jason Cong, Lei He, Cheng–Kok Koh and Patrick H. Madden, "Performance Optimization of VLSI Interconnect Layout", *Department of Computer Science, University of California*, Los Angeles, CA 90095, pps. 1–99.

* cited by examiner

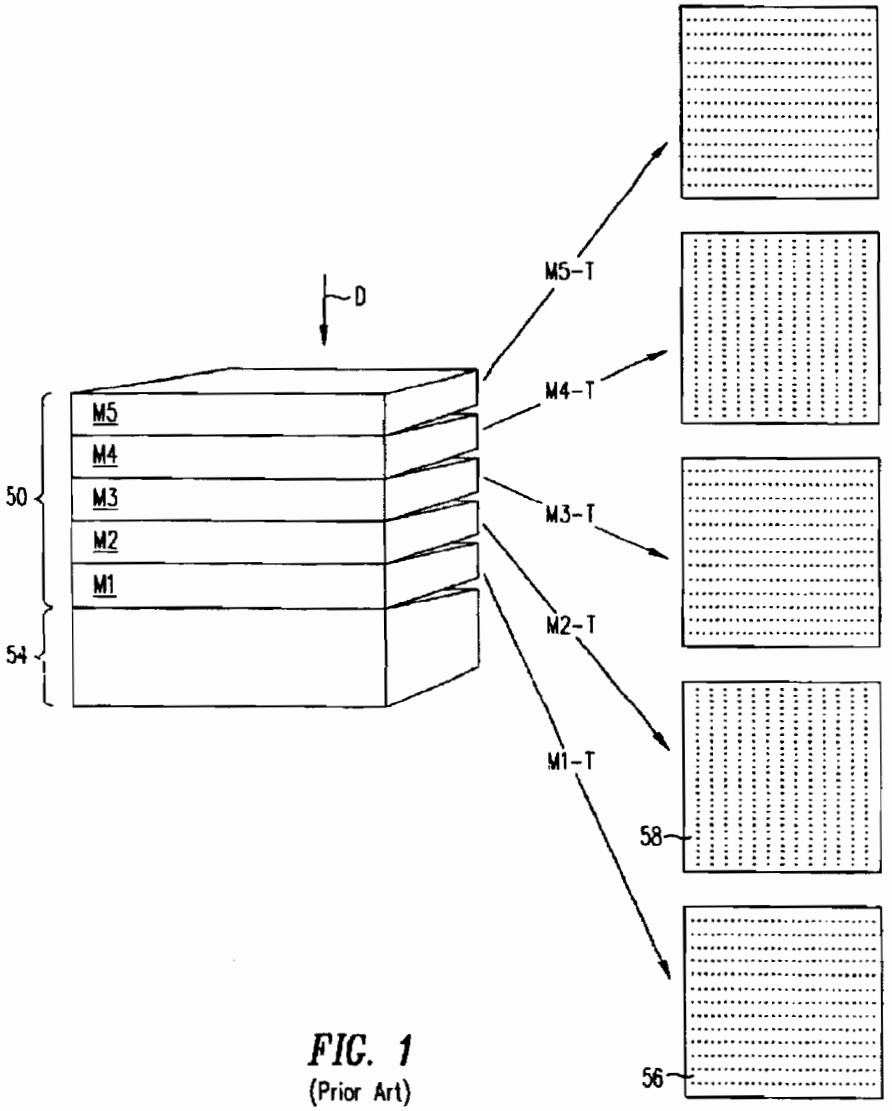


FIG. 1
(Prior Art)

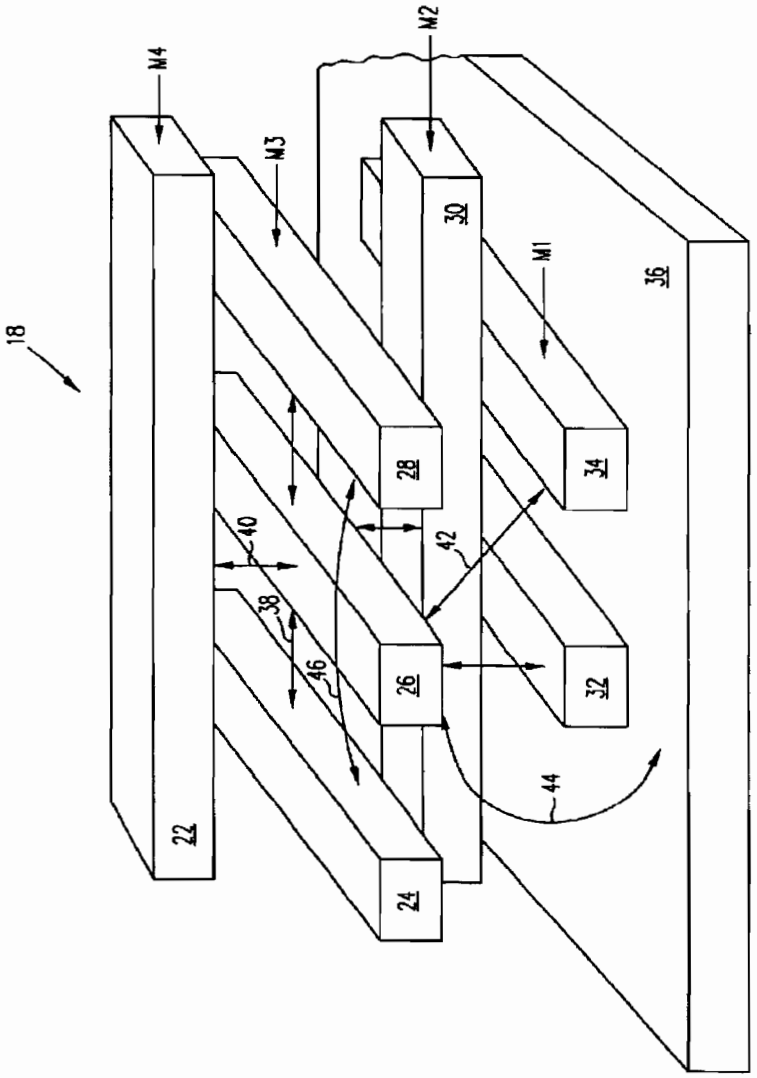


FIG. 2
(Prior Art)

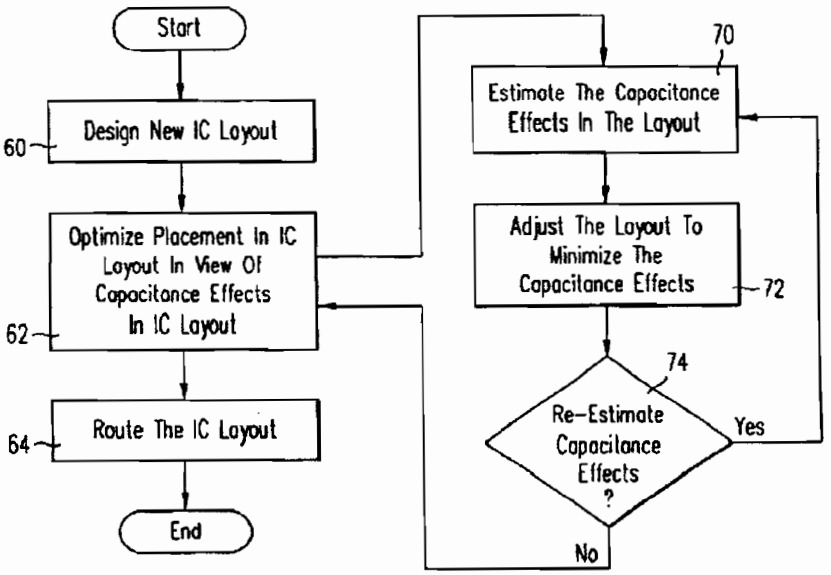

Start
Design New IC Layout
60
Optimize Placement In IC Layout In View Of Capacitance Effects In IC Layout
62
Route The IC Layout
64
End
70
Estimate The Capacitance Effects In The Layout
Adjust The Layout To Minimize The Capacitance Effects
72
74
Re-Estimate Capacitance Effects ?
Yes
No

FIG. 3

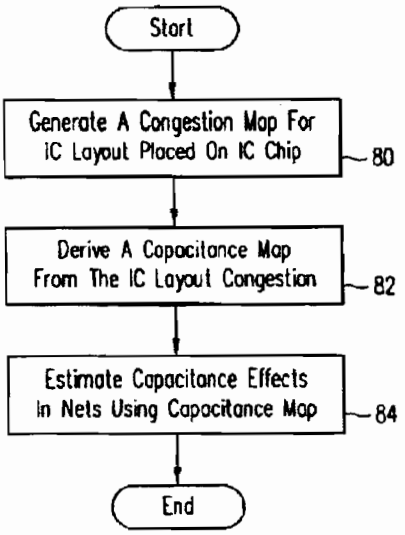

Start
Generate A Congestion Map For IC Layout Placed On IC Chip
80
Derive A Capacitance Map From The IC Layout Congestion
82
Estimate Capacitance Effects In Nets Using Capacitance Map
84
End

FIG. 4

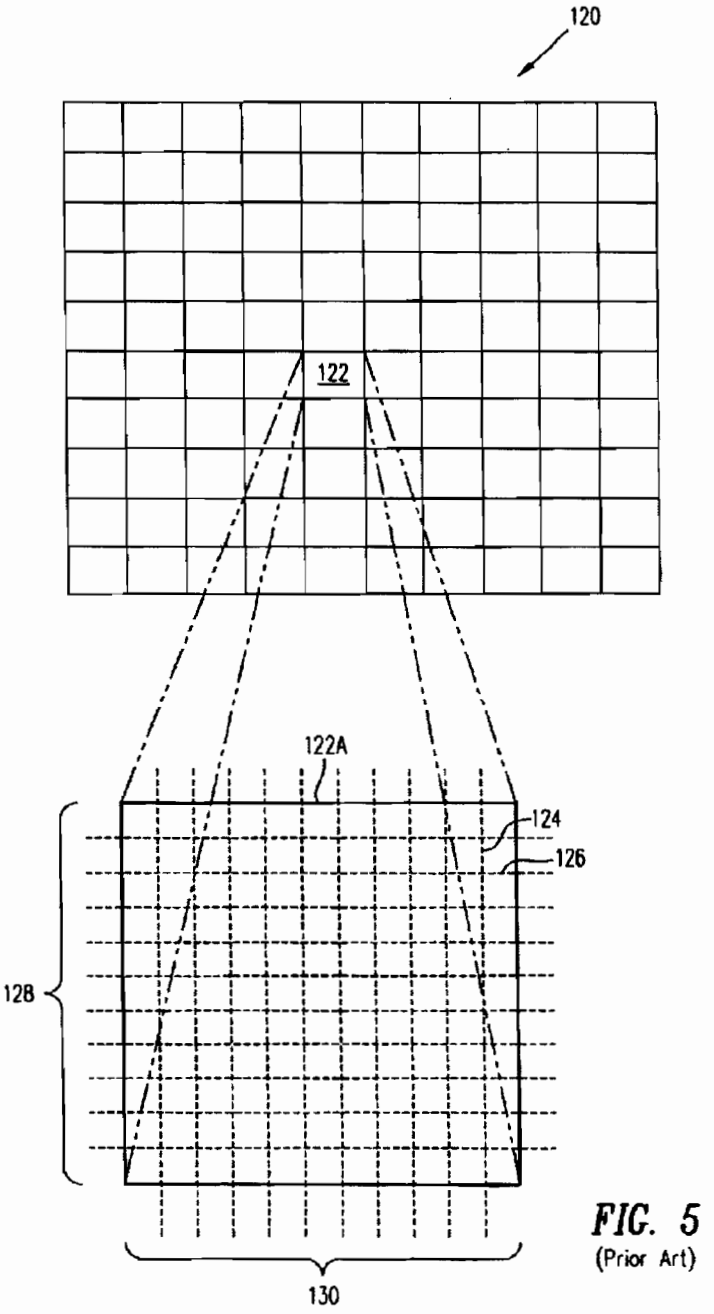


FIG. 5
(Prior Art)

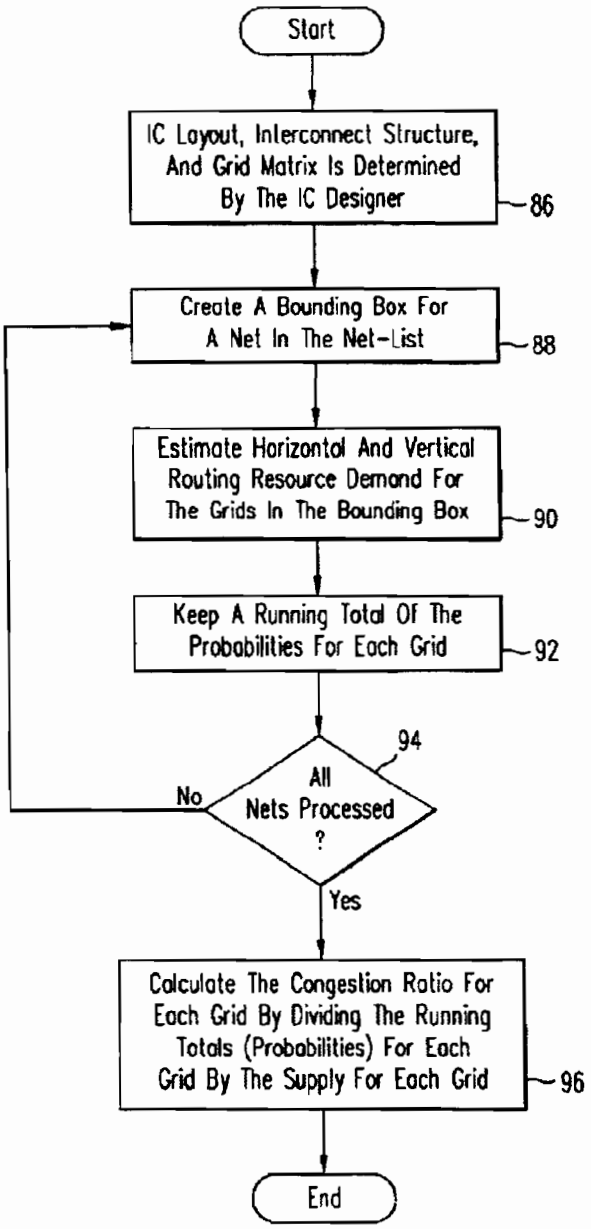

Start
IC Layout, Interconnect Structure, And Grid Matrix Is Determined By The IC Designer
86
Create A Bounding Box For A Net In The Net-List
88
Estimate Horizontal And Vertical Routing Resource Demand For The Grids In The Bounding Box
90
Keep A Running Total Of The Probabilities For Each Grid
92
94
All Nets Processed ?
No
Yes
Calculate The Congestion Ratio For Each Grid By Dividing The Running Totals (Probabilities) For Each Grid By The Supply For Each Grid
96
End

FIG. 6

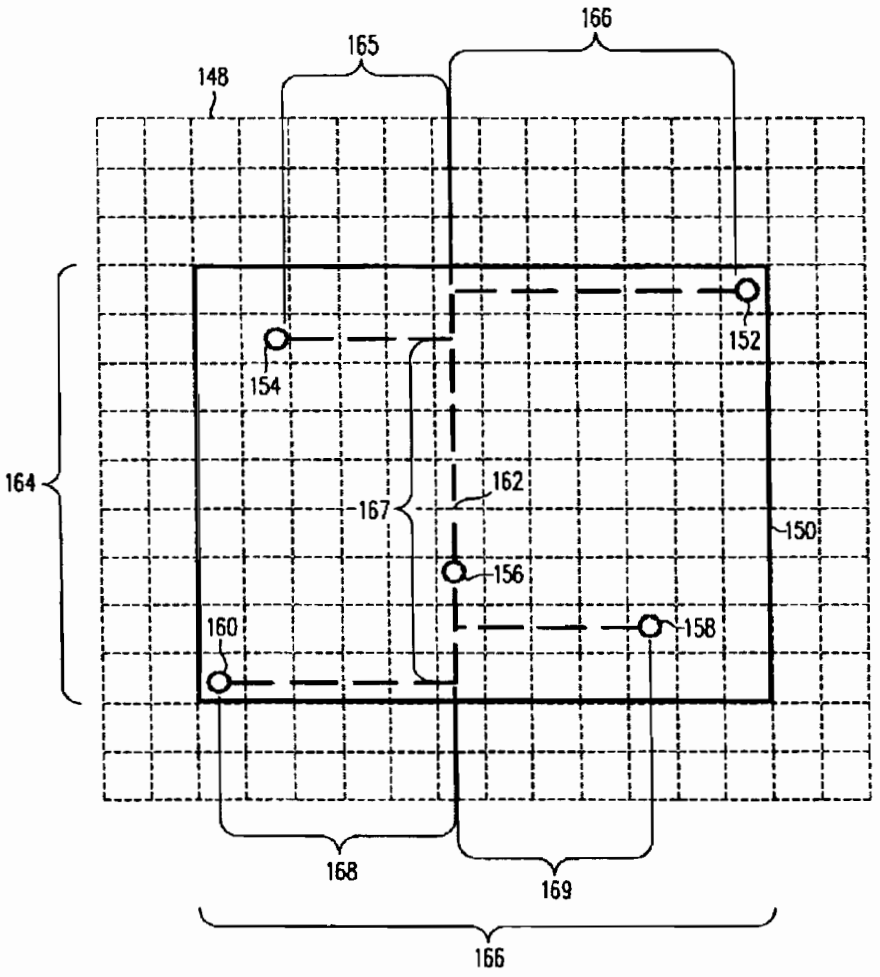


FIG. 7

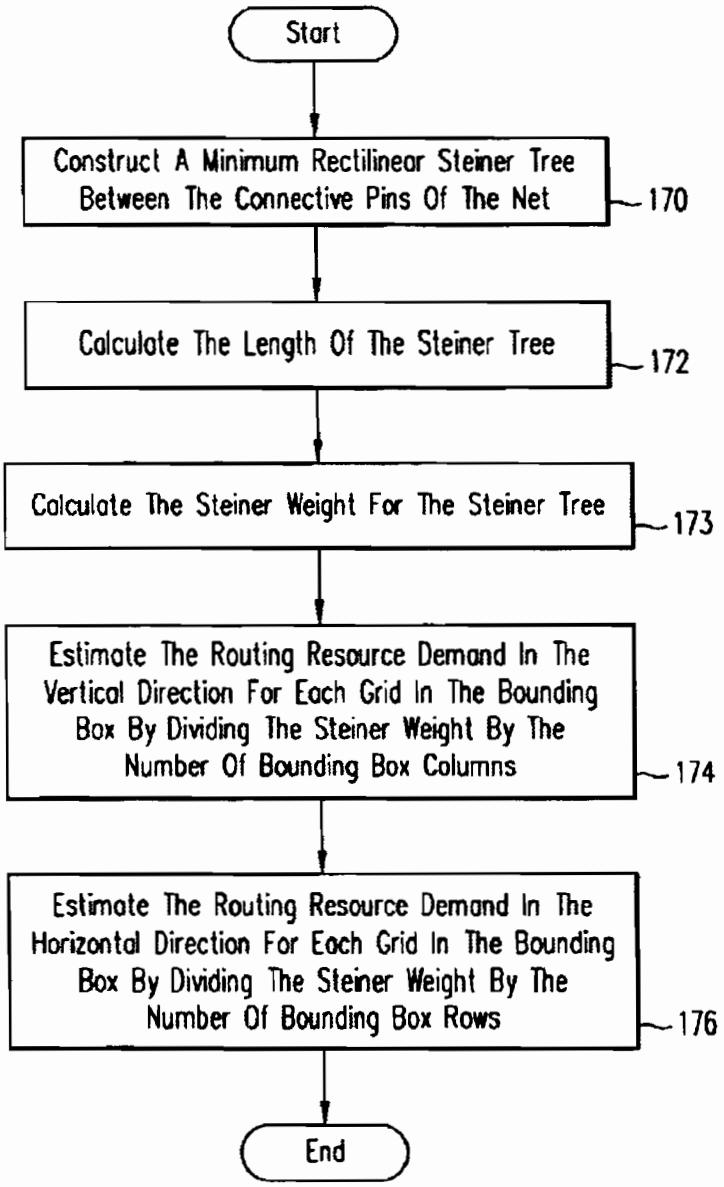

Start
Construct A Minimum Rectilinear Steiner Tree Between The Connective Pins Of The Net
170
Calculate The Length Of The Steiner Tree
172
Calculate The Steiner Weight For The Steiner Tree
173
Estimate The Routing Resource Demand In The Vertical Direction For Each Grid In The Bounding Box By Dividing The Steiner Weight By The Number Of Bounding Box Columns
174
Estimate The Routing Resource Demand In The Horizontal Direction For Each Grid In The Bounding Box By Dividing The Steiner Weight By The Number Of Bounding Box Rows
176
End

FIG. 8

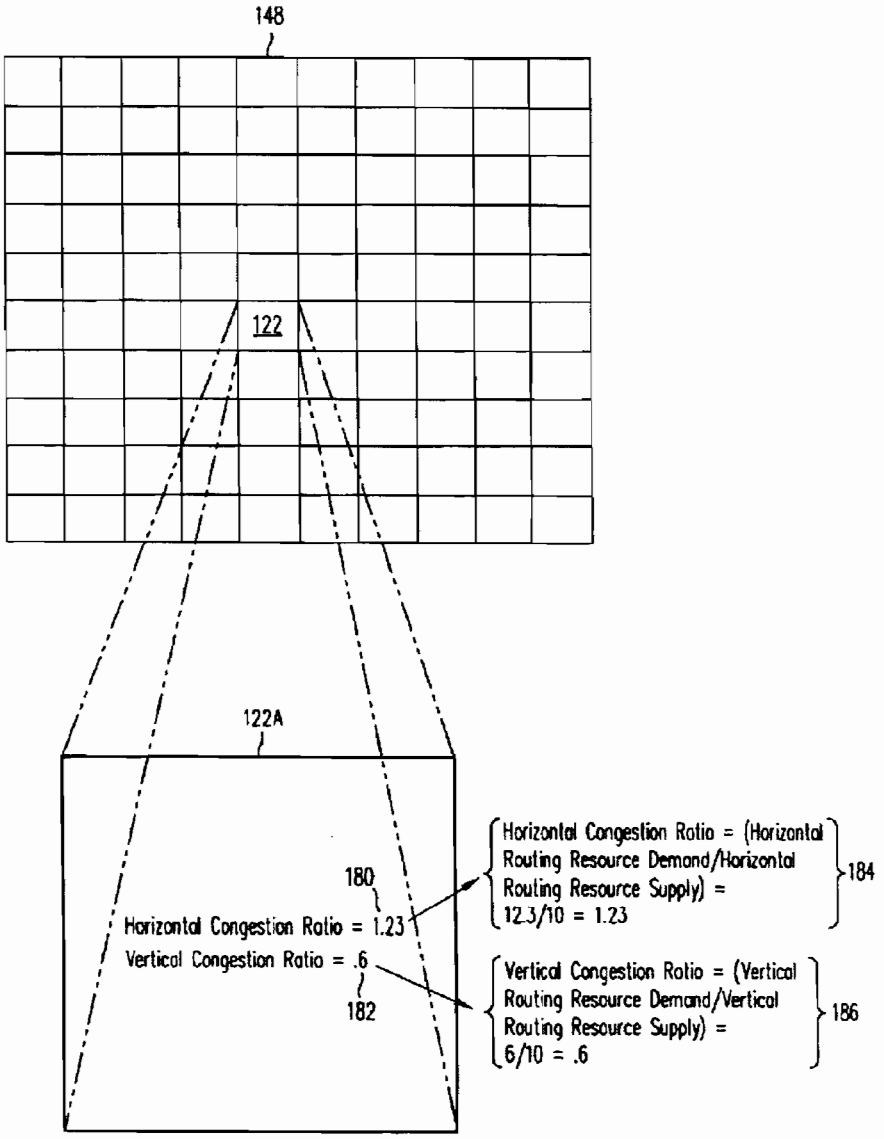

148
122
122A
180
Horizontal Congestion Ratio = 1.23
Vertical Congestion Ratio = .6
182
Horizontal Congestion Ratio = (Horizontal Routing Resource Demand/Horizontal Routing Resource Supply) = 12.3/10 = 1.23
184
Vertical Congestion Ratio = (Vertical Routing Resource Demand/Vertical Routing Resource Supply) = 6/10 = .6
186

FIG. 9

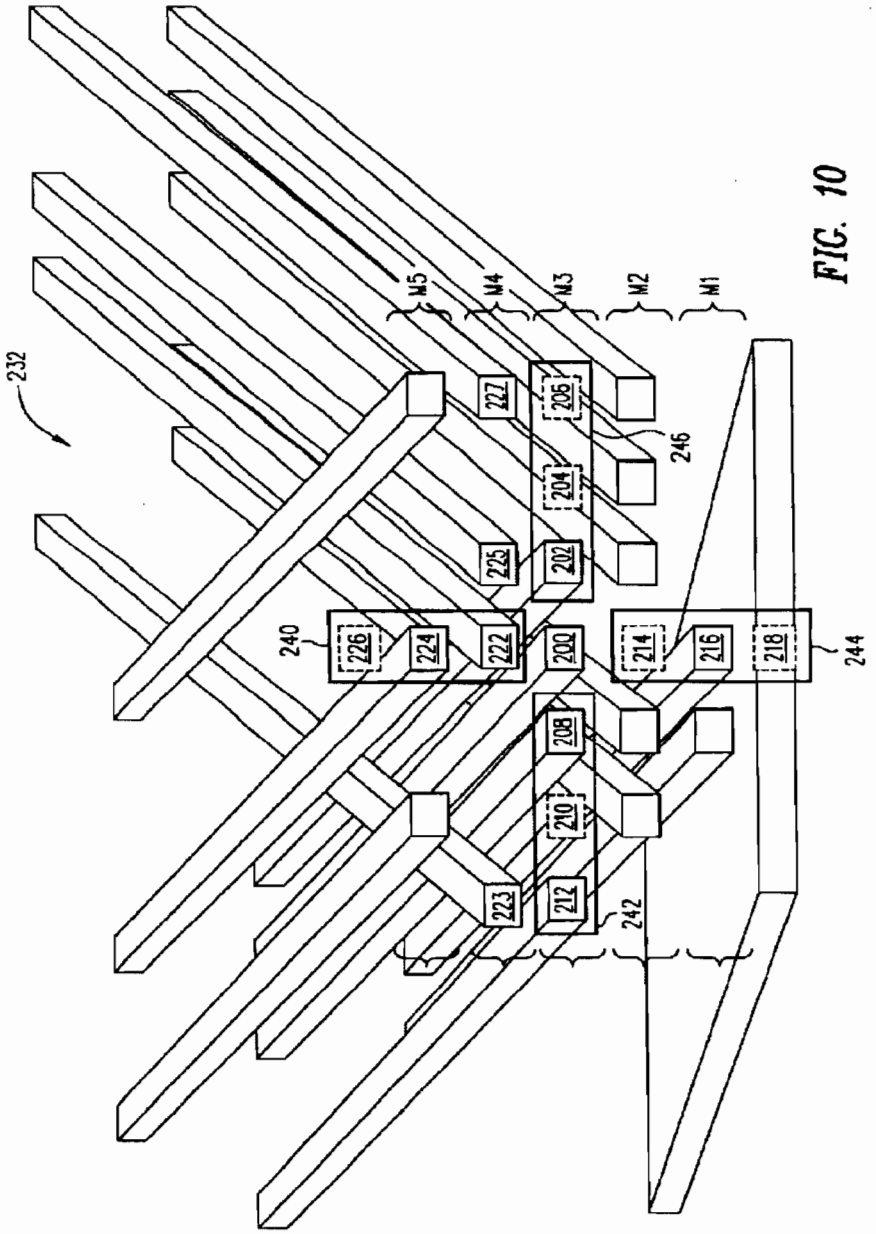

M5
M4
M3
M2
M1
232
227
206
246
204
225
202
240
226
224
222
200
214
216
218
244
208
210
223
212
242

FIG. 10

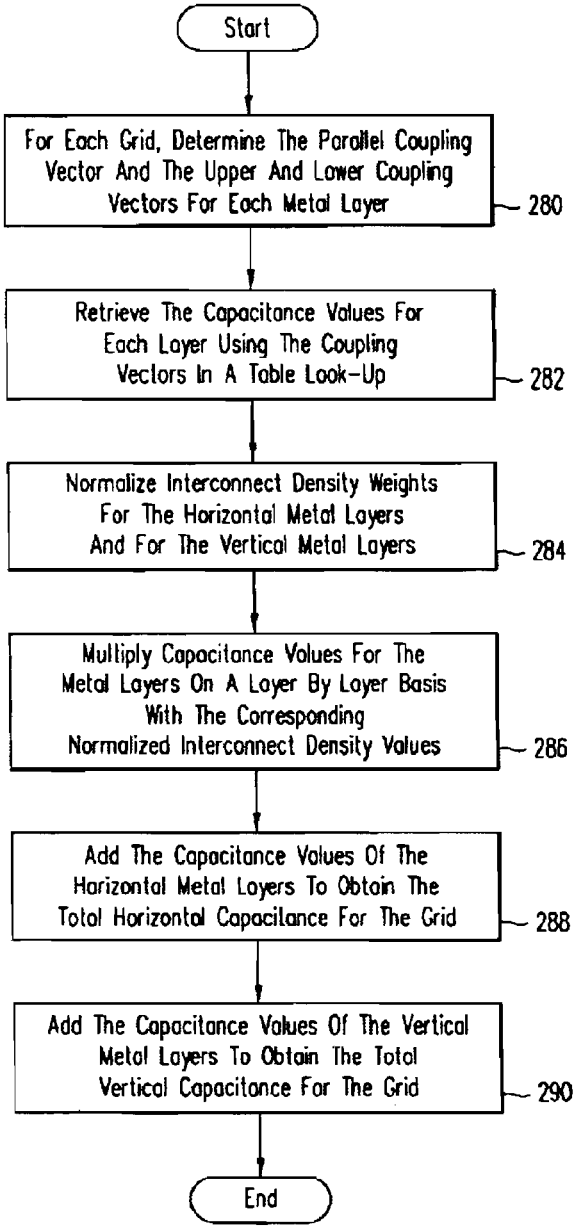

Start
For Each Grid, Determine The Parallel Coupling Vector And The Upper And Lower Coupling Vectors For Each Metal Layer
280
Retrieve The Capacitance Values For Each Layer Using The Coupling Vectors In A Table Look-Up
282
Normalize Interconnect Density Weights For The Horizontal Metal Layers And For The Vertical Metal Layers
284
Multiply Capacitance Values For The Metal Layers On A Layer By Layer Basis With The Corresponding Normalized Interconnect Density Values
286
Add The Capacitance Values Of The Horizontal Metal Layers To Obtain The Total Horizontal Capacitance For The Grid
288
Add The Capacitance Values Of The Vertical Metal Layers To Obtain The Total Vertical Capacitance For The Grid
290
End

FIG. 11

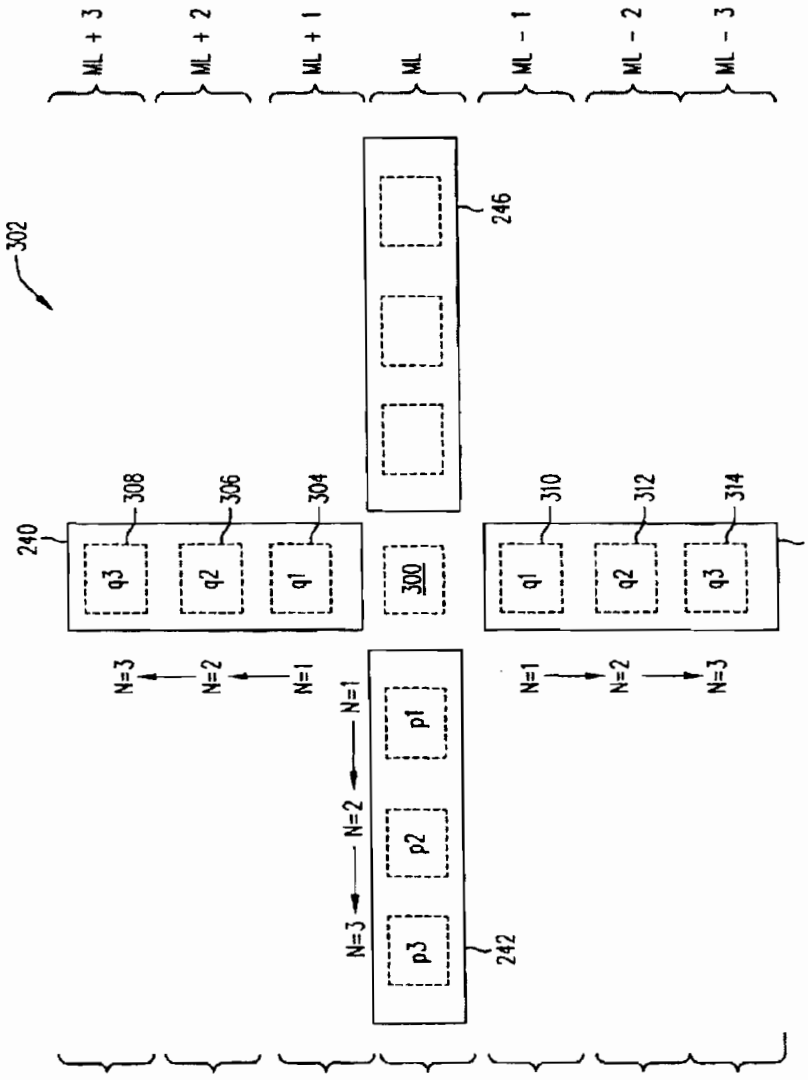

ML + 3
ML + 2
ML + 1
ML
ML - 1
ML - 2
ML - 3
302
246
240
308
306
304
q3
q2
q1
300
310
312
314
244
q1
q2
q3
N=3
N=2
N=1
N=1
N=2
N=3
N=1
N=2
N=3
p1
p2
p3
242

FIG. 12

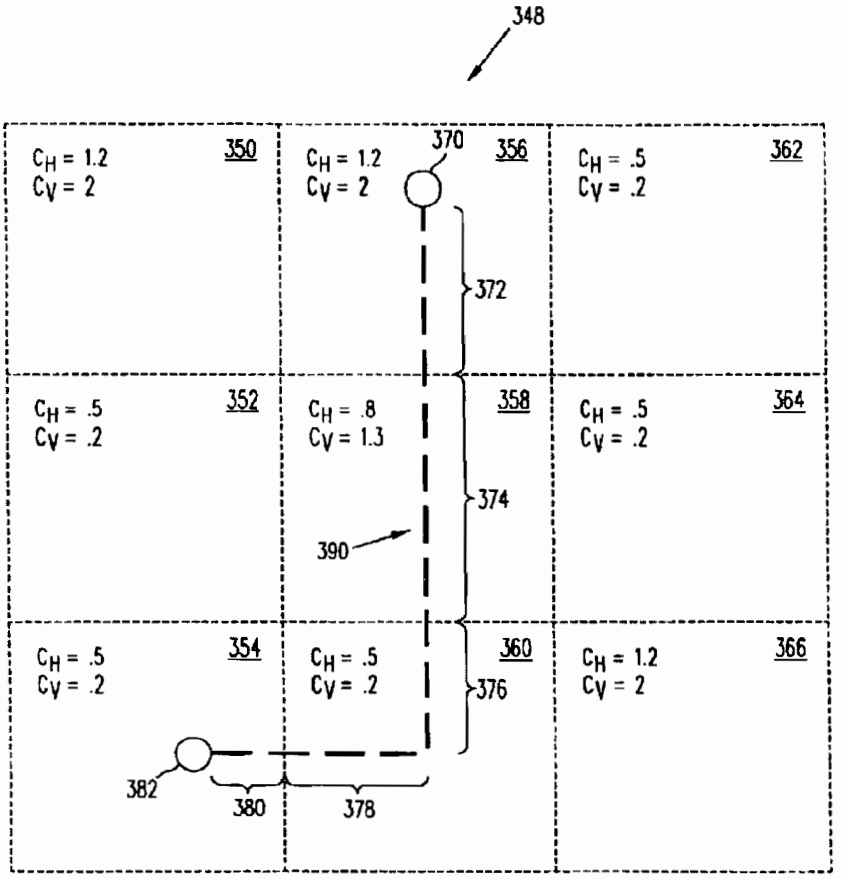

348
C_H = 1.2
C_V = 2
350
C_H = 1.2
C_V = 2
370
356
C_H = .5
C_V = .2
362
372
C_H = .5
C_V = .2
352
C_H = .8
C_V = 1.3
358
C_H = .5
C_V = .2
364
374
390
C_H = .5
C_V = .2
354
C_H = .5
C_V = .2
360
376
C_H = 1.2
C_V = 2
366
382
380
378

FIG. 13

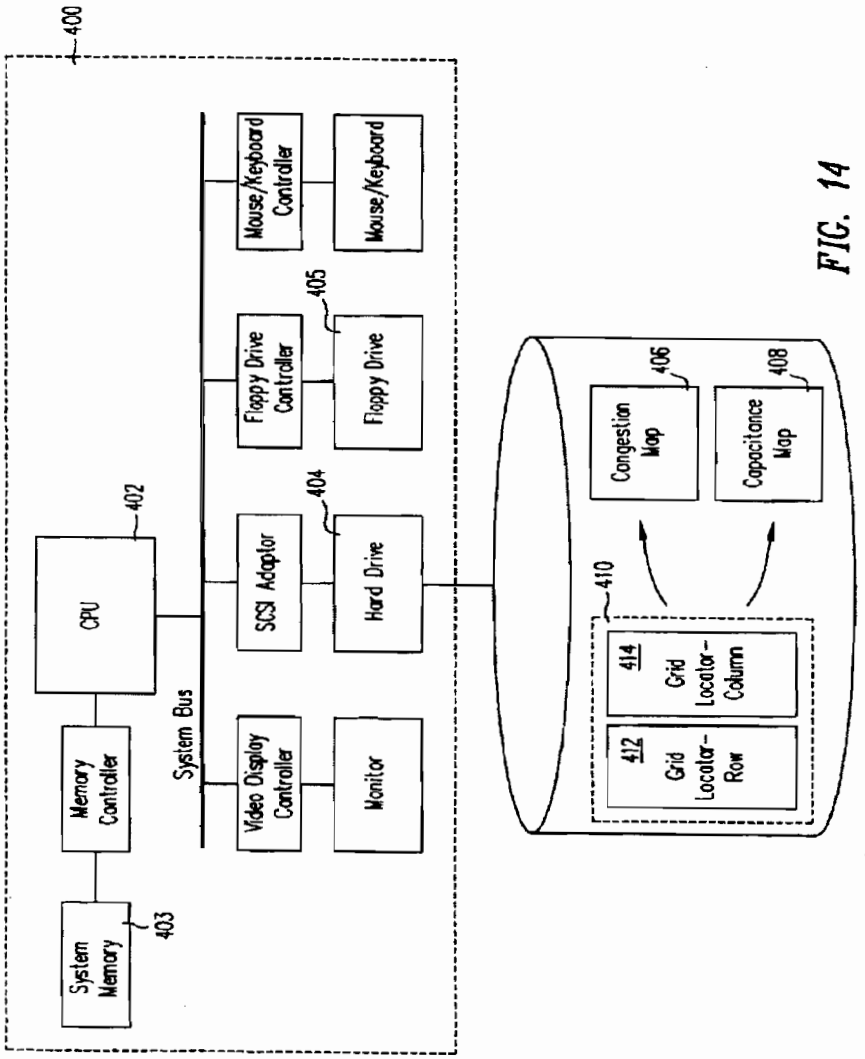

400
CPU
402
Memory Controller
System Memory
403
System Bus
Mouse/Keyboard Controller
Mouse/Keyboard
Floppy Drive Controller
405
Floppy Drive
SCSI Adaptor
404
Hard Drive
Video Display Controller
Monitor
410
414
Grid Locator-Column
412
Grid Locator-Row
Congestion Map
406
Capacitance Map
408

FIG. 14

ESTIMATING CAPACITANCE EFFECTS IN INTEGRATED CIRCUITS USING CONGESTION ESTIMATIONS

BACKGROUND

Throughout the 1990's to the present, IC design and fabrication has progressed to smaller and smaller scales. Recent developments, for example, include the design of "systems on a chip," or "SoCs," using VDSM ("Very Deep Submicron") technology. At VDSM dimensions, IC device sizes smaller than 0.25 microns (the sub-micron level, where one micron is one-millionth of a meter) are achieved. At such VDSM levels of integration, a number of design complexities arise. One of these is the well-known timing problems caused by capacitance effects generated within closely located conductive wires (hereafter, "interconnects") interconnecting the semiconductor devices (hereafter, "devices") constituting the integrated circuit. By way of background, a typical IC chip includes a semiconductor substrate containing upwards of a million devices of varying sizes (some devices of VDSM dimensions); the devices are in turn connected by interconnects running in metal layers layered atop the semiconductor substrate. A "metal layer structure" as used herein refers to the structure of metal layers **50** (FIG. **1**) within which the interconnects are to be routed to connect the devices embedded in the semiconductor substrate. Interconnects are typically routed parallel to one another within each metal layer as illustrated in FIG. **1**, although other configuration are sometimes used. In many common metal layer structures, the direction of the interconnects in adjacent layers is orthogonal. In addition, typical IC chips include a conductive substrate layer beneath the semiconductor substrate opposite to the metal layer structure that operates as a ground plane, as described next.

In FIG. **1**, a typical metal layer structure **50** having five metal layers is illustrated. (It should be noted that different IC designs may employ differently configured metal layer structures—e.g., a three-metal layer structure where interconnects are routed in the same track direction for each metal layer, or a six-metal layer structure where a single layer includes interconnects routed orthogonally to the interconnects in the other five layers.) As shown in FIG. **1**, five metal layers **50**—including layers **M1**, **M2**, **M3**, **M4** and **M5** respectively—are shown consecutively layered atop semiconductor substrate **54** (note that a ground plane beneath semiconductor substrate **54** is not shown). It should also be noted that the illustration in FIG. **1** is an abstraction, and that many tracks in the lower metal layers, e.g., **M1** and **M2**, may actually be unavailable for routing due to blockages caused by macrocells (i.e., large logic devices such as a CPU) protruding from the semiconductor substrate **54**. Each metal layer M1 through **M5** includes a set of parallel routing tracks that are illustrated as the dotted lines in M1-T through M5-T. M1-T through M5-T provide a top view of the parallel track structures (dotted lines) in metal layers **M1** through **M5**. (Some metal layer structures may have non-parallel routing tracks, and certain embodiments of the invention described below are not intended to be limited to any particular configuration of the routing tracks.) Thus, the routing tracks (**M1-T** for metal layer **M1** include routing track **56** running in a horizontal direction relative to the reader, when the reader views metal layer **M1** in the direction D (which is perpendicular to a plane of layer **M1**). Likewise, the routing tracks for **M2-T** include routing track **58** running in a vertical direction relative to the reader (i.e., orthogonal to the routing tracks in layers **M1** and **M3**). For purposes of this disclosure, metal layers will also be referred herein as "horizontal metal layers" or "vertical metal layers" depending on the routing track direction relative to the reader, unless otherwise noted.

Capacitance effects on a single interconnect **26** within a typical metal layer structure **18** may arise from a number of adjacent interconnects, as illustrated in FIG. **2**. In FIG. **2**, metal layer structure **18** includes a ground plane **36**, first metal layer **M1**, second metal layer **M2**, third metal layer **M3**, and fourth metal layer **M4**. (Note that the semiconductor device layer is not shown but is normally placed between ground plane **36** and first metal layer **M1**). **M1** in turn includes interconnects **32** and **34**; **M2** includes interconnect **30** running orthogonal to interconnects **32** and **34** in **M1**; **M3** includes interconnects **24**, **26** and **28** running orthogonal to interconnect **30**; and **M4** includes interconnect **22** running orthogonal to interconnects **24**, **26** and **28**, and parallel to, for example, interconnect **30** in **M2**. It should be noted that such prior art metal layer structures **18** may be varied in terms of interconnect direction (e.g., interconnects may run diagonally or otherwise in a metal layer), number of metal layers (e.g., two to eight layers), the order of metal layers (e.g., adjacent layers may have the same interconnect direction), and the number and location of ground planes (the first or a second ground plane may be located between metal layers three and four).

As illustrated in FIG. **2**, capacitance effects felt by interconnect **26** are shown by arrows, e.g., arrows **38**, **40** and **42** (other arrows are also shown but not labeled, and arrow **46** illustrates the capacitance effects generated between interconnect **24** and **28**). As shown, capacitance effects are generated within interconnect **26** by its neighboring interconnects **22**, **24**, **26**, **28**, **32**, **34** and **36**, and the ground plane **36**; the effects of capacitance flow in two directions (as illustrated by the arrows drawn as double-headed arrows), and thus interconnect **26** also causes capacitance effects on its neighboring interconnects **22**, **24**, **26**, **28**, **32** and **34**. The interconnects surrounding a given interconnect, e.g. **26**, whether a parallel interconnect in the same metal layer, e.g., **28** and **24** in **M3**, or an orthogonal interconnect in a different metal layer, e.g. **22** and **30** in **M4** and **M3** respectively, will be referred to as neighboring interconnects for purposes of this disclosure. In general, the effects of capacitance may be estimated as a function of the distance (among other things) between neighboring interconnects, and therefore increased integration—causing the interconnects to be routed in closer proximity to one another—increases the capacitance effects between the neighboring interconnects. Depending on how signals are transmitted within the neighboring interconnects at a given moment in time, the capacitance effects on a given interconnect may either delay or accelerate the signal carried by it. In a worst case scenario, the arrangement of interconnects in a particular metal layer structure may cause signals to be delayed beyond acceptable timing thresholds, resulting in arbitrary and recurrent chip malfunction.

Two prior art techniques for estimating capacitance are currently prevalent. In one technique, the IC designer relies upon his or her accumulated experience to estimate the potential capacitance effects in each new layout. Applicant notes that this method lacks accuracy because, for example, two layouts are typically different both in architecture and in scale of integration (IC designs tend to progress to smaller scales of integration). In another technique, a layout is first completely routed, then analyzed for capacitance effects to retrieve relevant statistical data for optimizing the layout in a second routing iteration. Applicant notes that this method has the drawback of requiring the layout to be routed at least

twice. In addition, Applicant notes that in typical applications, the prior art techniques use a single measure of capacitance to estimate capacitance for the entire design.

SUMMARY

In accordance with the present invention, a measure of congestion between interconnects of an integrated circuit (IC) chip being designed is estimated prior to routing of the interconnects through the chip. The estimated congestion for a portion of the chip is then used to derive an estimate of the capacitance effects caused in the portion by the presence of the interconnects. The capacitance effects are estimated as a function of the direction, the length, and/or the number of interconnects that are expected to traverse the portion. In some embodiments of the present invention, the estimate of capacitance effects is then used to optimize the placement of logical devices in a layout of the IC chip.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram illustrating an exemplary metal layer structure **50** for an IC design of the prior art.

FIG. **2** is a block diagram of the prior art illustrating the capacitance effects on a single interconnect **26** within a typical metal layer structure **18**.

FIG. **3** is a high-level flow diagram illustrating the general IC design production context in which the present invention may be used, according to some embodiments of the present invention.

FIG. **4** is a high-level flow diagram illustrating a method of performing stage **70** in FIG. **3**, according to prior art.

FIG. **5** is a block diagram illustrating a grid-matrix used to measure routing resources using the grid-matrix of FIG. **5**, according to some embodiments of the present invention.

FIG. **6** is a flow diagram illustrating a method of performing stage **80** in FIG. **4**, according to some embodiments of the present invention.

FIG. **7** is a block diagram illustrating a net in an exemplary grid-matrix **148** consisting of connective pins **152**, **154**, **156**, **158** and **160** interconnected to one another and circumscribed by a bounding box **150**, created by act **88** illustrated in FIG. **6**.

FIG. **8** is a flow diagram illustrating how the components in FIG. **7** are used to estimate routing resource demand in stage **90** of FIG. **6** using Steiner weights, according to some embodiments of the present invention.

FIG. **9** is a block diagram illustrating a congestion map derived in accordance with the flow diagram in FIG. **6**, according to some embodiments of the present invention.

FIG. **10** is a block diagram illustrating a coupling vector relative to a reference interconnect **200** in the post-routing metal layer structure of FIG. **2**, according to some embodiments of the present invention.

FIG. **11** is a flow diagram illustrating a method of estimating capacitance in the horizontal and vertical directions per grid of a grid-matrix, according to some embodiments of the present invention.

FIG. **12** is a block diagram illustrating the coupling vectors **240**, **242**, **244** and **246** computed to N=3 spacings from an estimated reference interconnect **300** at the pre-routing stage, according to some embodiments of the present invention.

FIG. **13** is a block diagram illustratiug a portion **348** of a resulting capacitance map and an exemplary two-pin net **390** positioned in the map, according to some embodiments of the invention.

FIG. **14** illustrates, in a block diagram, a grid factor **410** for use with a congestion map **406** and a capacitance map **408** by computer **400** in one particular implementation.

DETAILED DESCRIPTION

An "IC layout" for purposes of this disclosure refers generally to a computer-aided design of an integrated circuit. At the pre-routing stage relevant to this disclosure, the IC layout typically includes the proposed placement (location) of the semiconductor devices (hereafter "devices") on the IC semiconductor substrate, but not the actual manner in which the devices will be interconnected in the IC chip (i.e., in what tracks the router will route wires connecting the connective pins of the devices). Accordingly, the pre-routing IC layout typically includes a list of the devices in the IC design, a list of which connective pins belonging to the devices are interconnected (defining a "net"), and the placement of the devices (and their respective connective pins) on the IC semiconductor substrate. How the nets are actually routed (e.g., the length, direction and size of the interconnects as well as the metal layers used for routing the interconnects) is typically unknown at the pre-routing design stage. During the routing stage, the router uses a real-time routing algorithm that is generally too computationally intensive to run during the pre-routing stage. During the pre-routing stage, the IC layout may be optimized to preclude implementation failures arising during actual routing, e.g. by use of capacitance effects to change the placement of semiconductor devices prior to routing for purposes of avoiding timing anomalies caused by excessive concentration of capacitance effects generated by the interconnects.

Capacitance effects are estimated in certain embodiments of the present invention prior to the actual routing of a layout, using estimates of routing congestion of the to-be-routed interconnects. The congestion of such interconnects (for use in determining capacitance effects as described herein) may be estimated in any manner well known in the art, depending on the embodiment. For example, congestion may be estimated when a layout is optimized to avoid design placements that are not routable, or are inefficiently routable, due to an insufficient supply of interconnect resources in portions of the metal layer structure.

Although congestion problems may be solved by increasing the physical dimensions of the chip and/or the metal layer structure (adding metal layers)—thereby making additional interconnects available per unit area of the layout—this solution is generally undesirable because of the increased costs of producing larger-sized IC chips. Thus, typical prior art IC design tools enable congestion issues to be identified and avoided by appropriate placement of semiconductor devices during the pre-routing stage. See, for example, U.S. Pat. Nos. 5,847,965 and 5,798,936 that are incorporated by reference herein in their entirety. Interconnect congestion may be estimated as described in, for example, U.S. Pat. No. 5,587,923 that is also incorporated by reference herein in its entirety.

Interconnect congestion may also be estimated on a per unit area basis as distributed over a metal layer structure, as done by, e.g., the "Apollo" tool produced by Avant! Corporation of Fremont, Calif. For example, depending on the placement, a concentration of devices (and therefore connective pins) may be distributed over only certain portions of the IC chip. At these concentrated locations, the number of interconnects needed to connect the connective pins may exceed the available supply of interconnects supported by the metal layer structure; as used herein, the number of

interconnects needed per unit area of the layout shall also be referred to as the "routing resource demand." Routing resource demand may be measured in terms of tracks per unit area of layout; a track is a pre-determined line (also called "wire") in a metal layer structure along which a router is designed to route an interconnect. As used herein, "routing resource supply" shall refer to the number of tracks—and therefore interconnects—available to the router for routing; routing resource supply is typically measured on a per unit area basis over the metal layer structure.

In congestion estimation, the layout may be optimized to avoid congestion issues by adjusting the placement of the devices (and connective pins) so that concentrations of routing resource demand—especially in areas of relatively smaller supplies—may be re-distributed to less congested areas in the metal layer structure. A graphical representation of relative congestion distributed over the layout generated by an IC design tool may be used to obtain a congestion ratio per unit area of the layout (given a particular metal layer structure). As used herein, "congestion ratio" refers to the ratio of routing resource demanded by the layout divided by routing resource supplied in the metal layer structure on a per unit area basis. Such capacitance estimates may be used to optimize placement of devices in an IC layout.

So, any congestion estimation technique may be used in accordance with the invention to provide an efficient mechanism for estimating capacitance effects. Accordingly, in some embodiments of the present invention, a method and system for estimating capacitance effects is based on congestion estimation techniques of the prior art. An advantage provided by these embodiments is that existing congestion estimation technologies normally used for solving congestion problems are additionally used to solve timing problems caused by capacitance. In these embodiments, the resulting per-area congestion estimates obtained using congestion estimation techniques are in turn used to generate estimations of capacitance effects in corresponding areas in the interconnect structure.

In some embodiments of the present invention, a congestion map is created by individually estimating the congestion for all (or substantially all) portions of an IC chip. In such embodiments, the congestion map may include several portions (hereafter "grids") which together cover (or substantially cover) the surface of an interconnect structure; the estimate of congestion for each grid therefore provides a measure of the interconnect congestion in the metal layers of the interconnect structure below the grid and defined by the edges of the grid. In some embodiments, each grid covers a rectangular portion of the interconnect structure. Each such grid is defined by the intersection of a row and a column within a number of parallel rows and columns used to spatially represent the surface of the interconnect structure. In some embodiments, a congestion ratio is estimated for each grid by dividing the number of interconnects required in the grid by logical devices in a layout of the IC chip with the number of interconnects available for use in the interconnect structure defined by the grid. In some embodiments of the present invention, a congestion ratio is estimated for each grid for each direction in which the interconnects are designed to be routed through the grid; in some embodiments, the interconnects of a layer run along an x-axis and a y-axis (orthogonally) through each grid.

In some embodiments of the present invention, an estimate of the capacitance effects within a grid is derived from the congestion ratios estimated for the grid. In some embodiments, the capacitance effects are estimated using a predetermined table that enables a measure of the capacitance effect for a particular interconnect to be looked up as a function of, among other things, the probability of the particular interconnect having neighboring interconnects, i.e., other interconnects occupying routing tracks adjacent, above and below the particular interconnect. In some embodiments, an estimate of the probable existence of neighboring interconnects is derived for each grid from the congestion ratio estimates for the grid.

In some embodiments, the total capacitance effect exerted on an interconnect to be routed through multiple grids is estimated by determining the capacitance effects exerted within each grid through which the interconnect is to be routed, and then summing the resulting capacitance effects for each grid. In some embodiments, the capacitance effects exerted on an interconnect in a grid are estimated as a function of the length and the direction of the portion of the interconnect to be routed within the grid. The estimated total capacitance effects exerted on the interconnects to be routed in a particular IC design are then used, in some embodiments, to optimize the placement of the devices—and the interconnects connecting the devices—in the IC design for purposes of eliminating timing mistakes in the routed IC chip caused by excessive capacitance effects generated within the interconnect structure of the IC chip.

FIG. **3** is a high-level flow diagram illustrating a process for designing an integrated circuit, according to some embodiments of the present invention. In stage **60**, a new IC layout is designed by placement of semiconductor devices (also called logic devices). Next, in stage **62**, the placement of devices is optimized in the layout based on estimation of capacitance effects, which in turn is based on congestion estimates. In stage **64**, the semiconductor devices are interconnected by a router using interconnects to be formed in a metal layer structure during automated design of the IC chip.

Stages **70** through **74** (FIG. **3**) illustrate in more detail stage **62** for some embodiments of the present invention. In stage **70**, capacitance effects in the IC layout are estimated based on the placement of the logic devices in the layout, and prior to routing of the interconnects between the logic devices. In stage **72**, the placement of logic devices may be physically adjusted (from their locations determined in stage **60**) to avoid excessive concentrations of capacitance effects in the metal layer structure; such an adjustment may affect placement of 1–2% of the devices in the IC. In stage **74**, a new estimation of capacitance effects based on the newly adjusted device placement may be performed for purposes of accurately estimating capacitance effects in the newly adjusted device placement. The need to update the capacitance estimations for a particular device placement arises because the capacitance estimations are derived from the device placement; accordingly, after adjustment to the placement of the devices in response to a given capacitance estimation, the given estimated capacitances no longer provide an accurate estimation of capacitance for the adjusted device placement (because the capacitance estimations prior to update were derived from the original device placement by stage **60**). The estimated capacitance effects are thus re-derived from the adjusted device placement to accurately reflect capacitance in the adjusted device placement.

Re-estimating new capacitance effects for the adjusted device placement is particularly recommended if substantial adjustments to the original device placement are performed in stage **72**. Different criteria may be used to trigger a re-estimation of capacitance effects (a re-computation of the capacitance map) based on adjustments to the device placement. In some embodiments, for example, a re-estimation of capacitance may be triggered by a pre-determined number of

discrete adjustments to the device placement, e.g., between 1000–2000 changes to the device placement (a change to the device placement includes, e.g., the addition of a new net, the removal of or modification to a pre-existing net).

In other embodiments, a re-estimation may be triggered if a pre-determined percentage of the nets in the net-list of the IC design are adjusted during optimization. In some of these latter embodiments, if more than e.g. 5% of the nets in the net-list are adjusted during optimization, then re-estimation of capacitance in the adjusted device placement may be automatically triggered; if less than e.g. 2% of the nets are adjusted, then re-estimation may not be triggered. In yet other embodiments, re-estimation of capacitance may be triggered by the occurrence of other processes, such as updation of a congestion map used in the IC design process for estimating interconnect congestion in the device placement; in these embodiments, updation of a congestion map may automatically trigger re-estimation of capacitance in the IC design.

Stage **70–74** may be repeatedly performed to ensure an accurate measurement of capacitance effects for a particular device placement throughout the course of design optimization. In some embodiments, the IC designer establishes a particular timing constraint to be reached during optimization. The need to re-compute estimations of capacitance in response to adjustments to device placement for accurately designing, or the use of a pre-determined timing constraint, are generally decisions dependent upon the design goals of the particular IC designer. In some embodiments, however, a placement is adequately optimized without iteration of stages **70–74**.

FIG. **4** is a high-level flow diagram illustrating a method for performing stage **70** in FIG. **3**, according to some embodiments of the present invention. In stage **80**, a particular IC layout is processed to produce a congestion map for the whole layout from individual congestion estimates of each of a number of portions of the layout. In stage **82**, the congestion map is used to derive a capacitance map for estimating the local capacitance effects within the metal layer structure. Stage **80** may be skipped, and a capacitance map may be directly created from individual congestion estimates for each portion of the layout, without first generating a congestion map.

In such embodiments, for example, congestion may be estimated only locally for a grid for purposes of deriving an estimate of capacitance locally within the grid, but is otherwise discarded without creating a congestion map for the entire layout. In stage **84**, the capacitance map is used to estimate the capacitance effects in the nets comprising the IC layout. Congestion may be estimated using a grid-matrix **120** (FIG. **5**) to measure routing resources, according to some embodiments of the present invention. In particular, a close-up view **122**A of an exemplary grid **122** within grid-matrix **120** is illustrated. Grid-matrix **120** comprises a two-dimensional abstraction of interlocking grids placed on—and coextensive with—the top surface of a metal layer structure in an IC layout; the grids (or "areas") are defined by a network of parallel horizontal and vertical lines forming a checker-board pattern of non-overlapping squares. In some embodiments, each grid, e.g., grid **122**, is therefore used to represent the routing resource usage in a corresponding cross-section of the typically multi-layered metal layer structure. (Please note that a typical grid matrix **120** may have less or more grids than the number illustrated in FIG. **5**).

The close-up view **122**A of exemplary grid **122** illustrates a number of vertical and horizontal routing tracks, e.g., **124** and **126** respectively, available to a router for routing interconnects through the metal layer structure defined by grid **122** (the tracks are shown as dotted lines). During congestion estimation, the particular metal layer in which the tracks, e.g., **124** and **126**, may be routed in the grid **122** is ignored. Therefore, as illustrated, ten routing resources (tracks) are available to the router in both the horizontal direction **128** and the vertical direction **130**; more or less routing tracks may be available in the other grids in the grid-matrix **120**. The number of routing tracks available in a particular grid may vary depending upon, for example, the grid size, the number of routing layers in the metal layer structure, the interconnect density in the routing layers, and the number of routing resources blocked by large semiconductor devices, such as a CPU.

FIG. **6** is a flow diagram illustrating a method for performing stage **80** in FIG. **4**, according to some embodiments of the present invention. In stage **86**, an IC layout (including the net-list for the layout), metal layer structure (e.g., the number and ordering of metal layers) and grid matrix (e.g., the dimensions of the grids) are previously determined by the IC designer. In stage **88**, a bounding box is created for a net in the net-list. A "bounding box" herein refers to the minimum box formed using grids that encompasses all of the connective pins of the net, as illustrated in FIG. **7**. Connective pins are the pins belonging to the semiconductor devices to be interconnected as defined by the particular net. In stage **90**, for each grid in the bounding box, the number of interconnects required by the layout to be routed through each grid—an estimation of routing resource demand—is estimated in both the horizontal and vertical routing track directions. In some embodiments of the present invention, stages **88** and **90** are performed by calculating a minimum spanning tree or, in yet other embodiments, a rectilinear Steiner tree—both well-known calculations to those skilled in the art—for each net in the net-list. In stage **92**, the resulting estimated number of vertical and horizontal routing tracks needed for each net per grid are stored in a running total for each respective grid.

For embodiments that use a minimum spanning tree or a rectilinear Steiner tree to estimate interconnect demand, stage **92** is performed by keeping a running total of interconnects to be formed in each grid in each direction by each minimum spanning tree or rectilinear Steiner tree calculated in stages **88–90**. In stage **94**, all of the nets in the net-list are processed through stages **88** through **92**, thereby resulting in running totals for each grid representing the total number of routing tracks in the vertical and horizontal directions demanded by the particular net-list. In stage **96**, congestion ratios for each grid are determined by dividing the running totals for routing resource demand calculated in stages **90** through **94**—i.e., routing resource demand for each grid—by the routing resource supply for each grid. The resulting congestion ratios (total routing resource demand divided by total routing resource supply for each grid) forms a congestion map. In some embodiments the congestion map provides a mapping of relative congestion in the horizontal and vertical directions on a grid by grid basis over the metal layer structure. In certain embodiments, the congestion map may be displayed as a colored image to the IC designer, wherein the higher congestion ratios are shown with colors different from lower congestion ratios, to provide ready detection by a human of high-congestion areas in an IC design, for purposes of optimization.

Routing resource demand for an exemplary net consisting of connective pins **152**, **154**, **156**, **158** and **160** (FIG. **7**) placed in an exemplary grid-matrix **148** may be computed

using any spanning tree method well known in the art. A bounding box **150** defined by the aforementioned connective pins includes a number of columns **166**, and a number of rows **164**. The columns **166** and the rows **165** equal the length of the net **162** in grids in the horizontal and vertical directions respectively. FIG. **8** is a flow diagram illustrating how the components in FIG. **7** are used to estimate routing resource demand in stage **90** of FIG. **6** using Steiner weights, according to some embodiments of the present invention. A Steiner weight is generally a measure of the ratio between the length of a net and its size, and may be used advantageously to estimate routing congestion in a metal layer structure as described in FIG. **8**.

Turning to FIG. **8**, in stage **170**, a minimum rectilinear Steiner tree **162** is constructed between the connective pins, e.g., **152**, **154**, **156**, **158** and **160**, of a given net. A minimum rectilinear Steiner tree **162** as used herein refers to a minimum spanning tree of rectilinear shape connecting all of the connective pins of a net. Depending on the implementation, the minimum rectilinear Steiner tree may be estimated (and not actually determined) because of excessive inefficiencies and complexities associated with algorithms for determining the exact minimum rectilinear Steiner tree for a given net (note that there may be more than one solution for a given net). In stage **172**, the length of an estimated minimum rectilinear Steiner tree is calculated in terms of grid lengths. Estimation of length in stage **172** is illustrated in reference to FIG. **7**, where the length of minimum rectilinear Steiner tree **162** equals the length of the line segments **165**, **166**, **167**, **168** and **169**. The lines segments **165**–**69** equal, approximately, the sum of 3.5 (segment **165**), 6 (segment **166**), 8 (segment **167**), 5 (segment **168**), and 4 (segment **169**), equaling a total of 26.5 grid lengths.

In stage **173**, the Steiner weight for the Steiner tree is estimated. The Steiner weight in the vertical direction for all of the grids in the bounding box **150** is estimated in some embodiments by dividing the minimum rectilinear Steiner tree length calculated in stage **172** by the number of columns **166** plus the number of rows **164** comprising the bounding box **150**. In FIG. **7**, the number of columns **166** in the bounding box **150** equals 12, and the number of rows equals 9; therefore, in this example, the Steiner weight for each grid within bounding box **150** equals 26.5 divided by 21, or 1.206. The Steiner weight of 1.206 also provides a measure of resource congestion, hereafter "congestion ratio." In particular, a congestion ratio of 1.206 means that for purposes of estimation, over 20% as many routing tracks in the vertical direction are required than are available within the grids defined by bounding box **150**. In stage **176**—in like manner as stage **174**—the routing resource demand in the horizontal direction for all of the grids in the bounding box **150** is estimated by dividing the Steiner weight by the number of rows **164** in the bounding box **150**; for the exemplary net in FIG. **7**, this equals 1.206 divided by 9, or 0.134. It should be noted that for purposes of congestion estimation, the ratio of the lengths of the minimum rectilinear Steiner tree to the lengths of the bounding box edges provides the relevant Steiner weight for stages **174** and **176**; consequently, other units of measurement instead of grid lengths may be used so long as a consistent measure of ratio is preserved.

An alternative to calculating Steiner weights for stage **173** is illustrated in Table 1. In this embodiment, the estimated Steiner weight is computed merely as a linear function of the connective pin count "p" of the net. The first order and zero order coefficients of the linear function are derived experimentally before use in the pre-routing optimization stage, from a large sample of Steiner weights computed from randomly generated nets in accordance with stages **170** through **176** of FIG. **8**. The estimated linear functions for computing Steiner weights is provided in Table 1, according to some embodiments of the present invention.

TABLE 1

# of Connective Pins (p)	Estimated Steiner Weight
$p \leq 3$	1
$3 < p \leq 10$	$p * 0.068 + 1$
$10 < p \leq 20$	$p * 0.045 + 1$
$20 < p \leq 30$	$p * 0.035 + 1$
$30 < p \leq 40$	$p * 0.033 + 1$
$50 < p \leq 50$	$p * 0.030 + 1$
$p > 50$	3

As shown in Table 1, the coefficients of the linear function change for each multiple of ten connective pins; the two exceptions to this are for small nets with 0 to 3 connective pins, and large nets with greater than 50 pins. In the former case of 0 to 3 connective pins, the Steiner weight is 1, and in the latter case of more than 50 connective pins, the Steiner weight is capped at 3. For nets having connective pins between 3 and 50, the Steiner weight varies between 1 and 3.

FIG. **9** is a block diagram illustrating a resulting congestion map that is generated in accordance with the flow diagram in FIG. **6**. In FIG. **9**, a close-up view **122A** of exemplary grid **122** in grid-matrix portion **148** is illustrated. In stages **88** through **92** of FIG. **6**, a running total of horizontal and vertical routing resource demands for each net in the net-list is associated with each grid in the matrix-grid for the layout. In stage **96**, the resulting running totals represent an estimate of the number of interconnects needed by the layout on a grid-wide basis across the metal layer structure.

Thus, for example, in the example of FIG. **9**, the exemplary running total **180** for horizontal routing demand for grid **122** may be assumed to be 12.3 for illustrative purposes (shown in computation **184**); this means that an estimated 12.3 tracks are needed by the layout in grid **122** for the horizontal direction. In like manner, the vertical routing demand for grid **122** for the layout may be assumed for illustrative purposes to be 6 (illustrated in computation **186**). In addition, the total supply of tracks for grid **122** may be assumed to be 10 in both directions (illustrated in computations **184** and **186**). Given these assumptions, the congestion ratios for grid **122** are therefore the total routing demand divided by the total routing supply in the horizontal and vertical directions; in this example, the horizontal congestion ratio **180** is 1.23, and the vertical congestion ratio **182** is 0.6. The horizontal congestion ratio indicates that more routing tracks are needed than supplied for this layout in grid **122**, and the vertical congestion ratio indicates that an adequate number of routing tracks are available to implement the layout in the grid **122**. The horizontal and vertical congestion ratios are accordingly calculated for each grid, thereby providing a two-dimensional per-grid measure of resource congestion—a congestion map—for a particular layout and metal layer structure.

A capacitance map, as used herein, is analogous to a congestion map but provides a measure of capacitance effects—instead of congestion—as distributed across a metal layer structure, typically on a grid by grid basis. In some embodiments of the present invention, the capacitance map is derived from the congestion map. In other

embodiments, a capacitance map may be created without first creating a map of congestion across the entire chip; in these embodiments, for example, capacitance estimates may be derived from individual congestion estimates on a grid by grid basis, wherein the congestion estimates are not stored together to form a map. In yet other embodiments, capacitance may be estimated only in one or more localized areas of the metal layer structure, for example, in one or more grids where substantial congestion (and therefore capacitance) is predicted to occur. Those skilled in the art will recognize that the principles and teachings of the present invention may be embodied in these and similar embodiments.

In addition, some embodiments according to with the present invention may estimate capacitance using previously developed tools for estimating actual routing congestion that will be present at the routing stage. In particular, capacitance is often calculated at the routing stage using a conventional Table Look-Up Model in which capacitance values are retrieved for a given interconnect by performing a look-up call to a table of pre-computed capacitance values. The capacitance values in the table are pre-computed on the basis of a predetermined set of parameters associated with the geometry of the interconnects; the parameters may include the number of metal layers in the metal layer structure, the number and location of neighboring interconnects, and the spacing, width and pitch (width plus spacing) of the interconnects. (A "neighboring interconnect" and a "coupling vector" are described in more detail below.) Because the spacing, width and metal layers used in the metal layer structure are typically a matter of design choice known during the pre-routing stage, the Table Look-Up Model may be advantageously used during the pre-routing optimization stage if coupling vectors for an interconnect are estimated.

In some embodiments of the present invention, a method and system for estimating coupling vectors from congestion ratios are provided. As used herein, a "coupling vector" generally provides a measure of the probable existence of interconnects neighboring ("neighboring interconnects") a hypothetical interconnect located in a portion of the design, e.g., interconnect **200**; it should be noted that "coupling vector" is also used herein to describe the actual existence of neighboring interconnects in the post-routing metal layer structure shown in FIG. **10** (this usage of the term, however, is for illustrative purposes). A coupling vector for purposes of using the Table Look-Up Model is described in reference to FIG. **10** below.

FIG. **10** is a block diagram illustrating four coupling vectors **240**, **242**, **244** and **246** relative to interconnect **200** in the post-routing metal layer structure of FIG. **1**, according to some embodiments of the present invention. (It should be noted that the direction in which interconnect traverse in adjacent metal layers e.g. metal layers M5 and M4 in FIG. **10** is orthogonal—in accordance with FIG. **1**—even though FIG. **10** illustrates a different angle (smaller) e.g. between interconnects **222** and **224**, for purposes of simplicity of drawing.) The neighboring interconnects may be measured in two directions for capacitance effects—parallel and orthogonal—relative to interconnect **200**. Interconnects lying in the same metal layer as interconnect **200** are typically parallel to interconnect **200**; interconnects lying in other layers are typically parallel to or orthogonal with interconnect **200**. For example, in FIG. **10**, interconnects **212**, **208** and **202** are parallel interconnects lying in the same metal layer M3 as the reference interconnect **200**; interconnect **222** is an orthogonal interconnect lying in the metal layer M4 above the reference interconnect **200**; interconnect **224** is a parallel interconnect lying in the metal layer M5 above interconnect **200**; and interconnect **216** is a parallel interconnect lying in the metal layer M1 below interconnect **200**. It should be noted that in the pre-routing optimization context, interconnect **200** typically refers to an interconnect segment defined by the length of a grid (not illustrated in FIG. **10**).

Typically the neighboring interconnects included in a coupling vector being estimated in the pre-routing design context are assumed to be located up to a predetermined number (N) of spacings away from the reference interconnect (resulting in an N-dimensional coupling vector); in some embodiments, the value of N may be 3. Generally, greater accuracy in estimating coupling effects may be achieved using coupling vectors with larger N values; the increased accuracy, however, is gained at the expense of computational overhead. An N value of 3 is used in one embodiment, for providing an acceptable trade-off between accuracy and performance for ordinary estimation purposes under current technologies. FIG. **10** illustrates exemplary coupling vectors **240**, **242**, **244**, and **246** in reference to an exemplary interconnect **200**.

Exemplary (N=3) coupling vectors **240**, **242**, **244** and **246** are illustrated in reference to FIG. **10**, according to some embodiments of the present invention. Three kinds of coupling vectors—an adjacency coupling vector, e.g., **242** or **246**, an upper-layer coupling vector, e.g., **240**, and a lower-layer coupling vector, e.g., **244**, are computed for purposes of estimating the existence of neighboring interconnects. In particular, adjacency coupling vectors **242** and **246** estimate the likely existence (in the pre-routing context) or the actual existence (in the post-routing context) of neighboring interconnects directly adjacent to an interconnect **200**, in the same metal layer, e.g., M3, as interconnect **200**. An upper-layer coupling vector **240** estimates the likely existence (in the pre-routing context) or the actual existence (in the post-routing context) of neighboring interconnects in the metal layers between interconnect **200** and the "sky layer" **226**. (The "sky layer" is a term in the art describing the empty space above the interconnect structure **232** for purposes of capacitance estimation—it has a capacitance effect of zero.) And, lastly, a lower-layer coupling vector **244** estimates the likely existence (in the pre-routing context) or the actual existence (in the post-routing context) of neighboring interconnects in the metal layers between interconnect **200** and the ground plane **218** (in contrast to the sky layer **226**, the ground plane **218** has a capacitance effect of one).

Adjacency coupling vector **242**, for example, may be computed as (1, 0, 1), representing parallel interconnects **208** and **212** in tracks one and three spacings away from interconnect **200** respectively (this means, e.g., that the probability of interconnect **208** being located in the respective track is 1.0, or 100%). Upper-layer coupling vector **240** may likewise be represented as (1, 1, sky), representing interconnect **222** in metal layer M4, interconnect **224** in M5, and the sky layer **226**. It should be noted that a sky layer and a ground plane layer are computed as additional coupling dimensions in upper-layer coupling vector **240** and lower-layer coupling vector **244** respectively. It should also be noted that the upper and lower-layer coupling vectors may not include a sky layer dimension or ground plane layer dimension if the sky layer or ground plane layer are beyond N spacings from a given interconnect. For example, in FIG. **10**, if the coupling vectors are computed for an interconnect lying in metal layer M2, the vertical coupling vector above the reference interconnect only requires consideration of

interconnects in M3, M4 and M5 (where N=3), because the sky layer exerts miniscule capacitance effect; likewise, the vertical coupling vector for layers below the M2 reference interconnect would only contain two dimensions, M1 and the ground plane.

Although the metal layer structure of FIG. **10** is shown already routed, the embodiments of the present invention are directed to pre-routing optimizations of the metal layer structure. Accordingly, the physical locations of the interconnects as illustrated in FIG. **10** are as yet unknown; this includes the location of neighboring interconnects within the metal layer relative to a given interconnect, and the neighboring interconnects distributed across the other metal layers. Consequently, at the pre-routing stage, the coupling vectors **240**, **242**, **244** and **246** require estimation, for purposes of using a Table Look-Up Method to determine capacitance.

FIG. **11** is a flow diagram illustrating a method for estimating capacitance effects in the horizontal and vertical directions per grid of a grid-matrix, according to some embodiments of the present invention. The stages in FIG. **11** will be described in reference to FIG. **10** where appropriate. In stage **280**, for each grid, the coupling vectors are estimated for each metal layer; the coupling vectors include adjacency coupling vectors **242** and **246**, an upper-layer coupling vector **240**, and a lower-layer coupling vector **244**. The coupling vectors are derived from the congestion map as described below in reference to Tables 3 and 4, and FIG. **12**, in accordance with some embodiments of the present invention.

In stage **282**, the coupling vectors are used in a conventional Table Look-Up Method for estimating the capacitance effects for each grid. In the Table Look-Up Method, parameters pre-determined by the IC designer (and dependent, e.g., on the physical characteristics of the IC chip), including the coupling vectors, the interconnect range value N, the number of metal layers, and the width and spacing of the interconnects within the grid, are used to index a table of pre-computed capacitance values. The pre-computed capacitance effects in the Table represent estimations derived empirically by a particular IC manufacturer from similar IC designs for use in guiding the actual routing of the IC design. Such tables are well-known in the art of IC fabrication, and are typically generated by the semiconductor foundry of major IC manufacturers using substantial testing and calibrations based on the particular fabrication requirements of the IC manufacturer (e.g., the type of metal typically used in this interconnect structure, the number and ordering of metal layers in the interconnect structure, and the functional characteristics of the IC design).

In stage **284**, the interconnect density weights for the horizontal and vertical metal layers are normalized to 1. Normalization is performed separately for the horizontal and vertical metal layers. For example, normalization of the horizontal metal layers is performed by first summing the interconnect density weights for a given horizontal metal layer, and then dividing the interconnect density weights of that horizontal metal layer by the summed result. Normalization of the vertical metal layers is similarly performed. The interconnect density weights represent estimations of the interconnect density as distributed across each metal layer in the interconnect structure of a particular layout design; the weights are empirically derived from analysis of the interconnect densities of generally similar layouts using similarly layered interconnect structures. The recommended interconnect density weights as a function of the number of metal layers in the interconnect structure for some embodiments compatible with the present invention are provided in Table 2:

TABLE 2

Interconnect Density Weights

# Metal Layers	Metal Layer	Interconnect Density Weights
2	M1	0.5
	M2	0.5
3	M1	0.05
	M2	1.0
	M3	0.95
4	M1	0.05
	M2	0.30
	M3	0.95
	M4	0.70
5	M1	0.02
	M2	0.35
	M3	0.63
	M4	0.65
	M5	0.35
6	M1	0.02
	M2	0.30
	M3	0.63
	M4	0.45
	M5	0.35
	M6	0.25
7 or More	M1	0.04
	M2	0.85
	M3	1.0
	M4	1.0
	>M4	0.75

The interconnect density weights of Table 2—if normalized—thus represent an empirically-derived estimation of the percentage distribution of interconnects over the metal layers of the metal layer structure as a function of the number of metal layers in the interconnect structure. For example, in a typical structure having five metal layers, as shown in Table 2, the first metal layer is comparatively the least dense (M1=0.02) relative to the other metal layers (e.g., M2=0.35). One reason for the comparative difference in interconnect density weights between M1 and M2 is that in typical IC layouts employing a five metal layers, with the first metal layer being adjacent to the semiconductor substrate embedding the solid-state devices—is often substantially blocked by macrocells and other large devices protruding into the first metal layer from the semiconductor substrate. Accordingly, a large proportion of the first metal layer is made unusable by the router for routing interconnects.

In stage **284**, normalizing the horizontal metal layers requires normalization of the interconnect density weights for M1 (0.02), M3 (0.63) and M5 (0.35), which, in this example, are already normalized (i.e., 0.02+0.63+0.35=1.0). If, however, the horizontal metal layers contain layers M1 and M3 only, then M1 (0.02) and M3 (0.63) are normalized to 0.03 (0.02/(0.02+0.63)) and 0.97 (0.63/(0.02+0.63)) respectively. In like manner, the vertical metal layers are identified and normalized using Table 2. In stage **286**, the capacitance values for the metal layers retrieved in stage **282** are multiplied on a layer by layer basis with the corresponding normalized interconnect density values. In stage **288**, the capacitance values for each horizontal metal layer resulting from stage **286** are added together to give a single grid's capacitance for interconnects running in the horizontal direction. In stage **290**, the capacitance values for each vertical metal layer resulting from stage **286** are added together to give a single grid's capacitance for interconnects running in the vertical direction.

It should be noted that the grid-matrix used for capacitance estimation is typically dimensioned identically to the grid-matrix used for congestion estimation; however, the two grid-matrices may also be dimensioned differently, e.g. one of the grid-matrices may be a multiple of the other (e.g., the congestion grid-matrix has twice as many columns and rows as the capacitance grid-matrix). When dimensioned differently, a function is used to map congestion values from one grid map (e.g. congestion) to another grid matrix (e.g. capacitance). The mapping function may be e.g. linear, depending on the relation between the grid matrices. It should also be noted that the grid matrices used for purposes of congestion and capacitance estimation may include rows and/or columns of differing heights and lengths respectively, and an appropriate mapping function is used.

A method is used, e.g. with Tables 3 and 4 below, and described in reference to FIG. **12**, for estimating the coupling vectors in stage **280** of FIG. **10**, according to some embodiments of the present invention. In stage **280**, estimates of the coupling vectors are calculated prior to routing; the coupling vectors being estimated at this stage are a function of the probability of existence of neighboring interconnects relative to a reference interconnect, because at this stage actual existence is unknown. Note that, referring back to FIG. **10**, the actual (and not estimated) coupling vectors **240**, **242**, **244** and **246** are illustrated (to N=3 dimensions (i.e., N=3 spacings from the reference interconnect **200**) in a post-routing metal layer structure, because in FIG. **10**, the location of the neighboring interconnects, e.g., **222**, **224**, **212**, **208**, **202** and **216** is shown post routing for the purposes of illustration only, and no estimation is required when such interconnects are known to exist.

FIG. **12** is a block diagram illustrating the probability of coupling vectors **240**, **242**, **244** and **246** computed to N=3 spacings from an estimate of the presence of a reference interconnect **300** at a pre-routing stage, according to some embodiments of the present invention. At the pre-routing stage, the locations of interconnects neighboring a hypothetical reference interconnect **300** are unknown; the coupling vectors for (which itself is not known to exist) reference interconnect **300** are therefore estimated based on probabilities. A reference interconnect **300** is assumed to exist, passing through a grid of interest (which is one of a number of grids in the grid matrix that are (e.g. sequentially) visited). Accordingly, the adjacency coupling vectors **242** and **246** (where N=3) may be defined as (p1, p2, p3), where p1 is the probability of the existence of an interconnect at N=1 (one spacing from the hypothetical reference interconnect **300**), p2 is the probability of the existence of an interconnect at N=2 (and no interconnect at N=1), and p3 is the probability of the existence of an interconnect at N=3 (and no interconnects at N=1 and N=2).

Estimation of the probable existence of a given interconnect at N spacings greater than 1 (N>1) may be based on the probability of absence of any intervening interconnects between the given interconnect and the reference interconnect; this technique reflects the negligible capacitance effect that a parallel interconnect exerts from behind a second parallel interconnect that lies between it and the reference connect (in the same metal layer). In some embodiments of the present invention, a congestion map as described in reference to FIGS. **4** through **9** may be used during the pre-routing stage to estimate the coupling vectors (e.g., p1, p2 and p3) for each metal layer on a grid by grid basis. Assuming the congestion ratio (ratio of routing resource supply over routing resource demand) is p—wherein p is set to 1 if the congestion ratio exceeds 1—then given a reference interconnect, the probabilities p1, p2, p3 and p4 of the reference interconnect having an adjacent interconnect in the first (p1), second (p2), third (p3), and fourth (p4) spacing from the reference interconnect may be estimated using the formulae provided in Table 3.

TABLE 3

Adjacency Coupling Vector Estimation

Dimensions	Estimation	Description
N = 1	p1 = p	Probability of interconnect routed in adjacent track (one spacing from reference interconnect)
N = 2	p2 = (1 − p) p	Probability of interconnect routed in second track, but not in first track (two spacings from reference interconnect)
N = 3	p3 = (1 − p)(1 − p) p	Probability of interconnect routed in third track, but not in first track or second track (three spacings from the interconnect)
N = 4	p4 = 1 − (p1 + p2 + p3)	For wire coupling effect due to wire in fourth track away, and no wires in the first, second and third tracks (four spacings from interconnect)

A general formula for estimating the existence of neighboring parallel interconnects up to N−1 spacings away from a given reference interconnect is provided by $p^N=(1-p)^{N-1}p$; estimation at N spacings away is the difference between 1 and the sum of the probabilities (p1, p2 . . . p^N) calculated using the aforementioned general formula (illustrated by p4 in Table 3). Thus, where the adjacency coupling vector is estimated to N=3 spacings, the resulting 3-dimensional adjacency coupling vector (p1, p2, p3) will equal (p, (1−p)p, 1−(p+(1−p)p)). Note that the adjacency coupling vectors that estimate the probable existence of neighboring parallel interconnects to each side of a reference interconnect are identical as a matter of probability (in the pre-routing context), and therefore only one adjacency coupling vector needs to be calculated (as described above), and thereafter used for retrieving a capacitance value using a Table Look-Up Method.

Estimation of the upper-layer coupling vector **240** and the lower-layer coupling vector **244** is performed similarly to the adjacency coupling vector **242**. In FIG. **12**, upper-level coupling vector **240** computed to N=3 metal layers is illustrated relative to a reference interconnect **300**, according to some embodiments of the present invention. Estimated reference interconnect **300** lies in metal layer ML, and upper-level coupling vector **240** computed to N=3 extends upward through three metal layers, labeled ML+1, ML+2 and ML+3 respectively. It should be noted that ML+1, ML+2 and ML+3 may be either a metal layer, the sky layer, or non-existent. The latter (non-existent) situation arises, for example, when the reference interconnect **300** lies in the uppermost metal layer of the metal layer structure, e.g., the fifth metal layer in a five layer metal layer structure; in this situation, ML is the fifth metal layer, ML+1 is the sky layer, and M1+2 and M1+3 are non-existent. In like manner, the lower-level coupling vector may have only a single dimension consisting of the ground plane layer.

Assuming the vertical congestion ratio (ratio of routing resource supply over routing resource demand for the vertical direction) is q—wherein q is set to 1 if the congestion ratio exceeds 1—then given a reference interconnect, e.g., **300**, the upper-level coupling vector (q1, q2, q3) **240** estimated across N=3 metal layers ML+1, ML+2 and ML+3

may be computed in accordance with Table 4, according to some embodiments of the present invention.

TABLE 4

Upper- and Lower-layer Coupling Vectors

Dimensions	Estimation	Description
N = 1	q1 = MIN(1.0, z1 * s),	Value of layer coupling vector value for first layer away, wherein z1 = q if the layer track direction is orthogonal, and z1 = p if the layer track direction is parallel
N = 2	q2 = MIN(1.0, (1 − q1) z2 * s)	Value of layer coupling vector value for two layers away, wherein z2 = q if the layer track direction is orthogonal, and z2 = p if the layer track direction is parallel
N = 3	q3 = MIN(1.0, (1 − q1) (1 − q2) z3 * s)	Value of layer coupling vector value for three layers away, wherein z3 = q if the layer track direction is orthogonal, and z3 = p if the layer track direction is parallel
	q_{gp} = 1 − (sum of qN)	Value of layer coupling vector value lumped to ground plane layer
	q_{sky} = 0	Value of layer coupling vector value lumped to the sky layer

The formulas in Table 4 generally correspond to the formulas in Table 3. In Table 4, a shielding factor "s" is used to increase the layer coupling vector values in order to augment the capacitance values retrieved using the Table Look-Up Method. A recommended default value for the shielding factor "s" is 2; in general, the value of the shielding factor is derived empirically. One method for deriving "s" is by trial and error comparing the capacitance values estimated using, e.g., the method described in reference to Tables 3 and 4, and FIG. **12**, with the actual capacitance measured from the already-routed IC chip (i.e., the actually routed device design). The Table 4 formulas in effect use the congestion ratios of the non-reference metal layers (the metal layers that do not include the reference interconnect) to estimate the likelihood of an interconnect running directly adjacent (immediately above or below) to a reference interconnect, e.g., interconnect **200**, in successive layers from the reference interconnect.

The directly adjacent interconnects (in all layers) above and below a reference interconnect are hereafter referred to as "direct interconnects" for purposes of this disclosure. In addition to the capacitance effects exerted by the direct interconnects, however, other neighboring interconnects in the metal layers above and below the reference metal layer (e.g. interconnects within the same layer as neighbors of a direct interconnect) also exert a relevant capacitance effect on the reference interconnect. This situation may be illustrated in reference to metal layer **M4** in FIG. **10**, where neighboring interconnects **223**, **225** and **227** exert an appreciable capacitance effect on the reference interconnect **200** (in addition the effect of direct interconnect **222**). Depending on the example, a direct interconnect **222** may or may not be actually present, but is assumed to be present and its estimate of capacitance effect is computed for use with the shielding factor "s".

Referring to FIG. **12**, the formulas in Table 4 provide an estimation of the likely existence of direct interconnects to be routed by the router—in the case of upper-layer coupling vector **240**—in tracks **304**, **306** and **308**, and—in the case of lower-layer coupling vector **244**—in tracks **310**, **312** and **314**. Thus, the Table 4 formulas omit the capacitance effects exerted by neighboring interconnects in the metal layers other than the direct interconnects. Accordingly, use of the Table Look-Up Method to retrieve the capacitance value for a layer coupling vector estimated merely on the basis of the direct interconnects—e.g., **304**, **306** and **308** for upper-layer coupling vector—will result in an underestimation in the total capacitance effect exerted by the interconnects in metal layer **M4**. A shielding factor "s" is therefore used to compensate for this underestimation by increasing (as described above) the layer coupling vector values by a multiple of "s." In addition, the Table 4 formulas accommodate the capacitance effects exerted by the ground plane layer, and the absence of capacitance effects exerted by the sky layer above the topmost metal layer.

The use of the Tables 3 and 4 to estimate the adjacency and layer coupling vectors may be illustrated in reference to a grid in metal layers **M1** and **M2** in the typical five metal layer structure depicted in FIG. **1**. For this example, the congestion ratio for the horizontal direction in the grid (metal layers **M1**, **M3** and **M5**) is hypothetically assumed to be 0.8, and the congestion ratio for the vertical direction in the grid (metal layers **M2** and **M4**) is assumed to be 1.5 respectively; these assumptions are designated as H=0.8 and V=1.5 respectively. In addition, the shielding factor "s" is assumed to be 2, and the coupling vectors are estimated to N=3 spacings. Starting with **M1** (a horizontally directed metal layer), p is assigned H and q is assigned V; thus, p=0.8 and q=1 (q is set to 1 because the congestion ratio H (1.5) exceeds 1). For the adjacency coupling vector (**p1**, **p2**, **p3**), **p1**=0.8, **p2**=(1−0.8)0.8 and **p3**=1−(0.8+(1−0.8)0.8); therefore, the adjacency coupling vector (**p1**, **p2**, **p3**)=(0.8, 0.16, 0.04). For the upper-layer coupling vector (**q1**, **q2**, **q3**), **q1**=MIN(1, (1*2)=1, **q2**=MIN(1, (1−1) 1*2)=0, and **q3**=MIN(1, (1−1)(1−0)1*2)=0; therefore, the upper-layer coupling (**q1**, **q2**, **q3**)=(1, 0, 0). For the lower-layer coupling vector (**q1**, **q2**, **q3**), **q1**=ground plane layer=1, and **q2** and **q3** are non-existent), therefore the lower-layer coupling vector (**q1**, **q2**, **q3**)=(1, 0, 0).

Turning next to metal layer **M2** (a vertically directed layer), p is assigned V and q is assigned H; thus, **p=1** (p is set to 1 because the congestion ratio H (1.5) exceeds 1) and q=0.8. Switching the values of p and q while moving from **M1** to **M2** reflects the change in track direction between **M1** and **M2**. The reason for this is that the adjacency coupling vector measures the parallel interconnect routing probabilities relative to a reference interconnect, and not relative to the direction (horizontal or vertical) of the reference interconnect; the layer coupling vectors likewise reflect the congestion of interconnects in surrounding layers, and not the directions of interconnects in the surrounding layers. Accordingly, for the adjacency coupling vector (**p1**, **p2**, **p3**), **p1**=1, p2=1(1−1), and p3=1−(1−0); therefore, the adjacency coupling vector (**p1**, **p2**, **p3**) is (1, 0, 0). For the upper-layer coupling vector (**q1**, **q2**, **q3**), **q1**=MIN(1, (0.8*2)=0.64, **q2**=MIN(1, (1−0.64)1*2)=1, and **q3** =MIN(1, (1−0.64)(1−1)0.8*2)=0; therefore, the upper-layer coupling vector (**q1**, **q2**, **q3**)=(0.64, 1, 0). For the lower-layer coupling vector (**q1**, **q2**, **q3**), **q1**=MIN(1, 0.8*2)=0.64, **q2**=ground plane layer=1−0.64=0.36, and **q3** are non-existent, therefore the lower-layer coupling vector (**q1**, **q2**, **q3**)=(0.64, 0.36, 0). The p and q for remaining metal layers **M3** through **M5** are calculated in a similar manner.

Such coupling vectors are used to generate a map of capacitance. FIG. **13** is a block diagram illustrating a portion **348** of a capacitance map, and an exemplary two-pin net **390** positioned in the map, according to some embodiments of the invention. In FIG. **13**, nine grids (**350** through **366**) of

exemplary portion **348** of a capacitance map constructed in accordance with this disclosure is illustrated. Each grid has an associated C_H and C_V representing the estimated total capacitance effects C for interconnects (or segments of interconnects) passing through the grid in either the horizontal "H" or vertical "V" directions. The capacitance effects C_H and C_V are a function of the length of the interconnect segment passing through the grid; thus, for example, for an interconnect that passes entirely through a grid in the vertical direction, the entire capacitance effect C_V is estimated to effect the interconnect segment. Estimating the capacitance effect in a net therefore includes determining the grids traversed by a net, and summing the respective capacitance effects associated with the particular interconnect segment lengths and directions in each of the aforementioned grids; this method is illustrated in reference to net **390** in capacitance map portion **348**.

As illustrated in FIG. **13**, net **390** consists of a horizontal segment and a vertical segment that traverses grids **354**, **360**, **358** and **356**. The horizontal segment further consists of horizontal segment **380** in grid **354** and horizontal segment **378** in grid **360**. The vertical segment further consists of vertical segment **376** in grid **360**, vertical segment **374** in grid **358**, and vertical segment **372** in grid **356**. The total capacitance effect C is estimated as the sum of the segment portions in each grid. Thus, traversing the net from connective pin **382** to connective pin **370**, the first line segment consists of horizontal segment **380** that traverses approximately 30% of the horizontal distance of grid **360**. The capacitance effect C_H for horizontal segment **380** is estimated accordingly as 0.3 multiplied by the C_H for grid, which is e.g. 0.2; thus, the C_H for horizontal segment **380** is estimated as (0.3)(0.2) or 0.06 (rounding numbers). In like manner, horizontal segment **378** traverses approximately 60% of grid **360**, and therefore the C_H for horizontal segment **378** equals 0.6 multiplied by the C_H for grid **360**, which is also e.g. 0.2; thus, the C_H for horizontal segment **378** is estimated as (0.6)(0.2) or 0.12.

Again, in like manner, vertical segment **376** traverses approximately 50% of grid **360**, and therefore the C_V for vertical segment **376** equals 0.5 multiplied by the C_V for grid **360**, or 0.5; thus, the C_V for vertical segment **378** is estimated as (0.5)(0.5) or 0.25. Vertical segments **374** and **373** are calculated similarly; thus, the C_V for vertical segment **374** equals (1)(1.3)=1.3, and the C_V for vertical segment **372** equals (0.7)(2)=1.4. The total capacitance for net **390** therefore equals the sum of the capacitances for each of the aforementioned horizontal and vertical segments **380**, **378**, **376**, **374** and **372**, or 0.06+0.2+0.25+1.3+1.4=2.96. In some embodiments, the unit of measurement for capacitance is specified by the IC designer, and is typically in pico-farads (10^{-12} farads) or femto-farads (10^{-15} farads). When the estimated capacitance exceeds the specification, appropriate steps may be taken by the router to reduce the capacitance (e.g. by routing through grids of lower congestion).

The method of the present invention may be performed in either hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present method may be carried out by software, firmware, or microcode operating on a general-purpose computer or computers or microprocessors of any type. FIG. **14** is a block diagram illustrating the principal hardware components of a general-purpose computer **400** compatible with some embodiments of the present invention. In particular, computer **400** includes at least one processor **402**, a system memory (e.g., RAM or ROM) **403**, and at least one computer-readable storage medium **404** (e.g., accessed by hard drive **405**). In some embodiments of the present invention, computer-readable storage medium **404** is encoded with a data structure **408** representing a capacitance map for a particular IC design.

In some of these embodiments, capacitance map **408** is a two-dimensional array of records for storing and accessing the capacitance effects estimated for grids of a grid-matrix in accordance with the methods described in this disclosure. In some embodiments compatible with the present invention, the dimensions of the two-dimensional array representing the capacitance map correspond to the dimensions of the relevant grid-matrix, thereby facilitating access to the estimated capacitance effects for the grids (because the capacitance value for a grid is located at a position within the two-dimensional array corresponding to the position of the grid within the grid-matrix). In some embodiments compatible with the present invention, each record in the capacitance map array comprises two real numbers for storing the capacitance effects in the vertical and horizontal directions for each grid.

In yet other embodiments compatible with the present invention, computer-readable storage medium **404** is additionally encoded with a data structure **406** comprising a congestion map **408**. In some of these embodiments, congestion map **408** is represented as a two-dimensional array of records for storing and accessing the congestion ratios estimated for the grids of a grid-matrix in accordance with the methods described in this disclosure. In some embodiments compatible with the present invention, the dimensions of the two-dimensional array representing the congestion map correspond to the dimensions of the relevant grid-matrix; this correspondence between the grid-matrix and congestion map facilitates access to the estimated congestion ratios for a particular grid because the capacitance value is located within the two-dimensional array in a position corresponding to the position of the grid within the grid-matrix. In some embodiments compatible with the present invention, each record in the capacitance map array comprises two real numbers for storing the congestion ratios in the vertical and horizontal directions for each grid.

In yet other embodiments compatible with the present invention, computer-readable storage medium **404** is yet further encoded with a data structure **410** for mapping the locations of the connective pins of the nets in the interconnect structure of the IC design to the corresponding grids within the grid-matrix. In some of these embodiments, each connective pin of a logical device is represented by a row number and a column number in a coordinate system defining locations on the interconnect structure of the IC design, a technique conventionally used in the art. In contrast to the grids of a grid-matrix, the rows and columns used to mark locations of the connective pins within the aforementioned coordinate system define areas of much finer granularity within the interconnect structure; a single grid thus includes multiple possible pin locations within the grid (not illustrated). Accordingly, in some embodiments of the present invention, the relevant congestion ratios and capacitance effects corresponding to various portions of the net defined by a set of connective pins are determined by identifying the relevant grids in which the portions of the nets are located; pin-to-grid data structure **410** is then used to map the portions of the net, including the connective pins, to the relevant grids. In some embodiments, data structure **410** includes two one-dimensional arrays **412** and **413**, wherein a first array **412** provides a mapping of a connective pin location (row) to a corresponding row of the congestion map **406** and capacitance map **408**, and a second array **413**

provides a mapping of a connective pin location (column) to a corresponding column of the congestion map **406** and capacitance map **408**. It should be noted that those skilled in the art will recognize that numerous, differing data structures may be used to implement different embodiments of the present invention. Moreover, the data structures may be encoded on the same storage medium that holds a description of the circuitry to be formed in the IC chip (e.g. in Verilog or VHDL).

Software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable medium, e.g., **404–406** (for example, ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD). Accordingly, the present invention is not limited to any particular computing platform.

While particular embodiments of the present inventions have been shown and described, it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications as fall within the true spirit of this invention.

I claim:

1. A computer-implemented method for estimating capacitance effects in interconnects to be routed in an integrated circuit, using a design simulating the integrated circuit prior to the routing, the design comprising a plurality of metal layers in an area, the method comprising:

retrieving from memory, capacitance effects for the area in each metal layer;

retrieving from memory, interconnect densities for each metal layer;

normalizing the interconnect densities for each metal layer; and

multiplying the normalized interconnect density for each metal layer with the retrieved capacitance effect for the area in the corresponding metal layer.

2. The method of claim **1** wherein metal layers sharing a common interconnect direction are normalized as a group.

3. The method of claim **2** further comprising:

summing the products of said act of multiplying, for metal layers sharing a common interconnect direction.

4. The method of claim **1** further comprising:

summing the products of said act of multiplying, for metal layers sharing a common interconnect direction.

5. The method of claim **1** wherein the area corresponds to a grid in a grid-matrix.

6. The method of claim **1** wherein said act of multiplying is repeated for each of a plurality of areas substantially covering the design.

7. The method of claim **1** wherein the capacitance effects are retrieved from a database of predetermined values, using a probability that a reference interconnect has neighboring interconnects.

8. The method of claim **7** wherein:

the neighboring interconnects are determined as coupling vectors.

9. The method of claim **8** wherein:

the coupling vectors comprise an adjacency coupling vector, a lower-layer coupling vector and an upper-layer coupling vector.

10. The method of claim **1** further comprising:

determining the probability using estimates of congestion among interconnects.

11. The method of claim **10** wherein:

the determining is performed for neighboring interconnects routed within a predetermined number of tracks, in the same metal layer of the design, from the reference interconnect.

12. The method of claim **10** wherein:

the determining is performed for neighboring interconnects routed within a predetermined number of metal layers from the reference interconnect.

13. A computer-readable storage medium encoded with a computer program, wherein the computer program comprises instructions for performing the acts of retrieving, normalizing and multiplying recited in claim **1**.

14. A computer system for estimating capacitance effects in interconnects to be routed in an integrated circuit, using a design simulating the integrated circuit prior to the routing, the design comprising a plurality of metal layers in an area, the computer system comprising:

means for retrieving from memory, capacitance effects for the area in each metal layer;

means for retrieving from memory, interconnect densities for each metal layer;

means for normalizing the interconnect densities for each metal layer; and

means for multiplying the normalized interconnect density for each metal layer with the retrieved capacitance effect for the area in the corresponding metal layer.

15. The computer system of claim **14** further comprising:

means for summing the products of said means for multiplying, for metal layers sharing a common interconnect direction.

16. The computer system of claim **15** wherein the capacitance effects are retrieved from a database of predetermined values, using a probability that a reference interconnect has neighboring interconnects, the computer system further comprising:

means for determining the probability using estimates of congestion among interconnects.

17. The computer system of claim **14** wherein the capacitance effects are retrieved from a database of predetermined values, using a probability that a reference interconnect has neighboring interconnects, the computer system further comprising:

means for determining the probability using estimates of congestion among interconnects.

18. The computer system of claim **14** further comprising:

the memory which is encoded with values of capacitance effects.

19. The computer system of claim **18** wherein:

the values of capacitance effects are held in a two-dimensional array;

each element in said two-dimensional array maps to a corresponding area covered by an interconnect structure; and

at least one value for each element in said two-dimensional array represents a capacitance effect generated within the corresponding area.

20. A computer-implemented method for estimating capacitance effects in interconnects to be routed in an integrated circuit using a design that simulates the integrated circuit prior to the routing, the method comprising:

retrieving from memory, interconnect densities for each metal layer;

normalizing the interconnect densities, for each metal layer;

determining a probability that an interconnect hypothetically routed in a metal layer of the design has neighboring interconnects in an area;

using the probability obtained from said determining to retrieve from memory capacitance effects for each metal layer in said area;

multiplying the normalized interconnect density for each metal layer, with capacitance effect for said area in the corresponding metal layer; and

summing the products of said act of multiplying for metal layers that share a common interconnect direction.

21. The method of claim **20** wherein:

the act of determining the probability uses estimates of congestion among interconnects.

22. The method of claim **20** wherein said act of multiplying is repeated for each of a plurality of areas substantially covering the design.

23. The method of claim **20** wherein:

the neighboring interconnects are determined as coupling vectors.

24. The method of claim **23** wherein:

the coupling vectors comprise an adjacency coupling vector, a lower-layer coupling vector and an upper-layer coupling vector.

* * * * *

TAB 19

US006857116B1

(12) **United States Patent**
Dahl et al.

(10) **Patent No.: US 6,857,116 B1**
(45) **Date of Patent: Feb. 15, 2005**

(54) **OPTIMIZATION OF ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN**

(75) Inventors: **Peter Dahl**, Cupertino, CA (US); **Byron Dickinson**, San Jose, CA (US); **Margie Levine**, Menlo Park, CA (US); **Paul Rodman**, Palo Alto, CA (US)

(73) Assignee: **Reshape, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days

(21) Appl. No.: 09/714,722

(22) Filed: Nov. 15, 2000

(51) Int. Cl.[7] G06F 17/50

(52) U.S. Cl. 716/12; 716/13; 716/14

(58) Field of Search 716/2, 4, 7, 8, 716/9, 11, 12, 13, 14

(56) References Cited

U.S. PATENT DOCUMENTS

4,890,238	A	12/1989	Klein et al.	364/491
5,187,671	A	2/1993	Cobb	364/490
5,309,370	A	5/1994	Wong	364/490
5,533,148	A	7/1996	Sayah et al.	382/240
5,544,088	A	8/1996	Aubertine et al.	364/489
5,576,969	A	11/1996	Aoki et al.	364/491
5,757,658	A	5/1998	Rodman et al.	364/491
6,243,854	B1	6/2001	Lavin et al.	716/19
6,425,113	B1 *	7/2002	Anderson et al.	716/5
6,618,849	B2	9/2003	Teig et al.	716/12

FOREIGN PATENT DOCUMENTS

WO	WO00/67163	11/2000	G06F/17/50

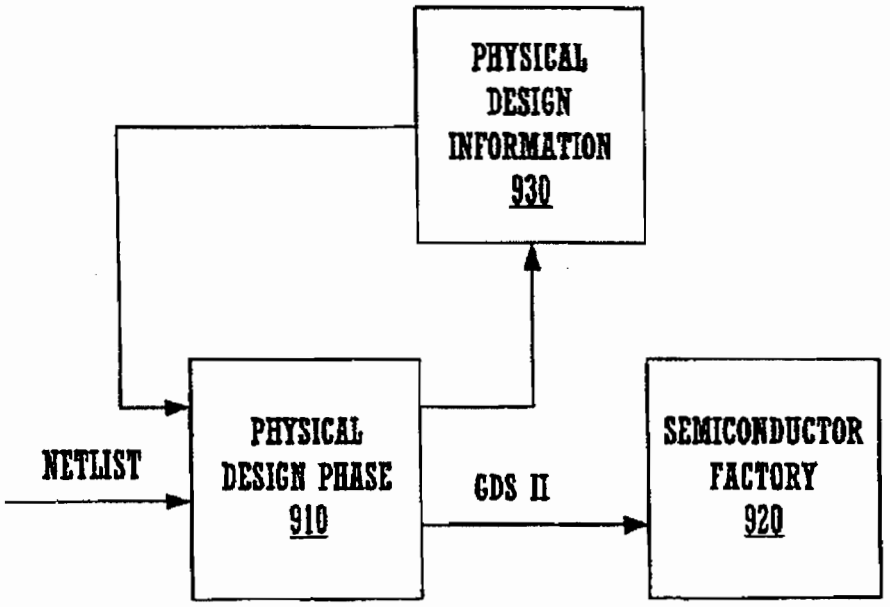
* cited by examiner

Primary Examiner—Thien F Tran
(74) *Attorney, Agent, or Firm*—Wagner, Murabito, & Hao LLP

(57) **ABSTRACT**

An abutted-pin hierarchical physical design process is described. The abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design and provides additional advantages and benefits. In particular, the abutted-pin hierarchical physical design does not have channels. Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced significantly.

52 Claims, 31 Drawing Sheets



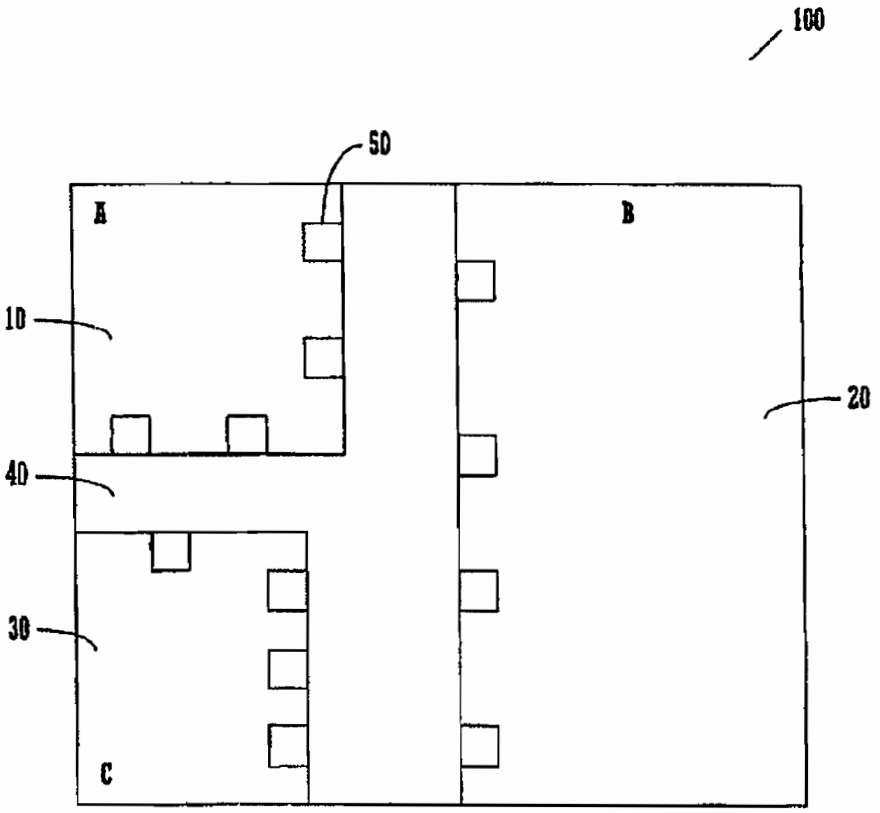


FIGURE 1
(Prior Art)

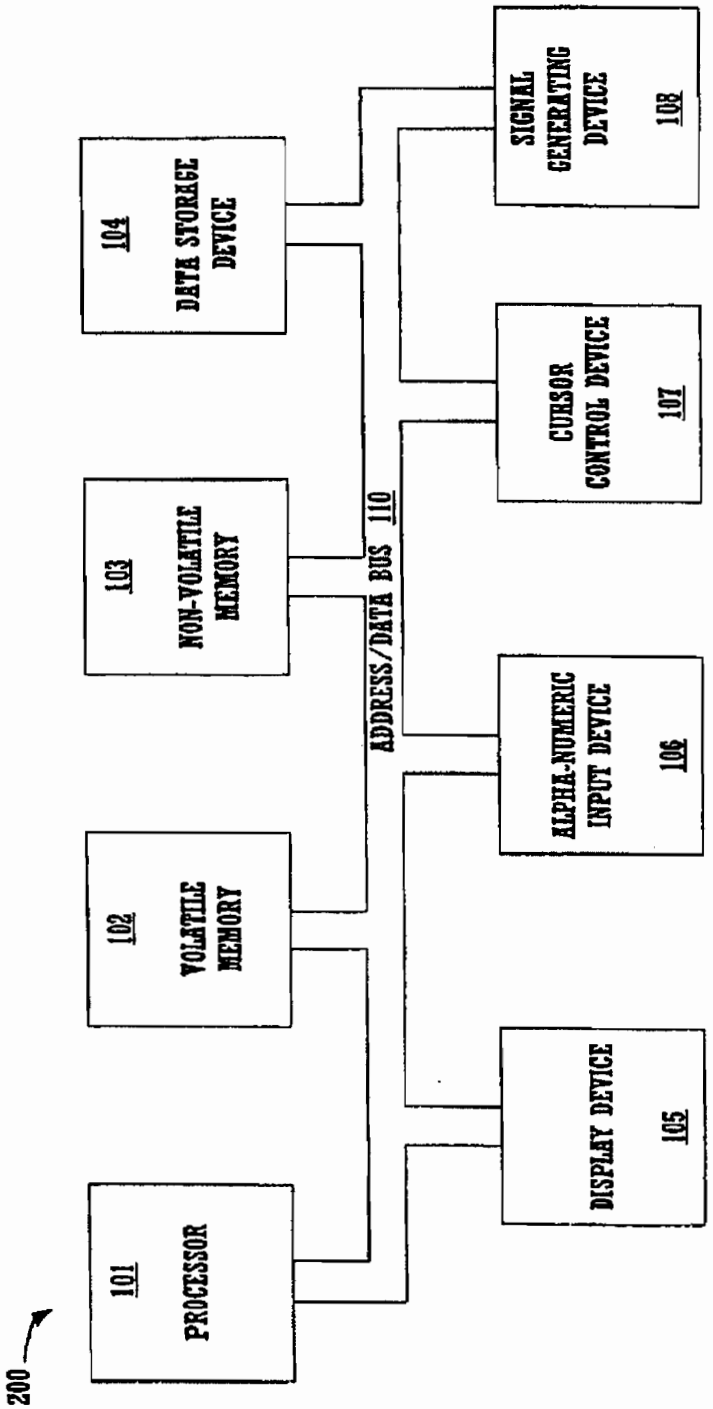

200
101 PROCESSOR
102 VOLATILE MEMORY
103 NON-VOLATILE MEMORY
104 DATA STORAGE DEVICE
ADDRESS/DATA BUS 110
SIGNAL GENERATING DEVICE 108
CURSOR CONTROL DEVICE 107
ALPHA-NUMERIC INPUT DEVICE 106
DISPLAY DEVICE 105

FIGURE 2

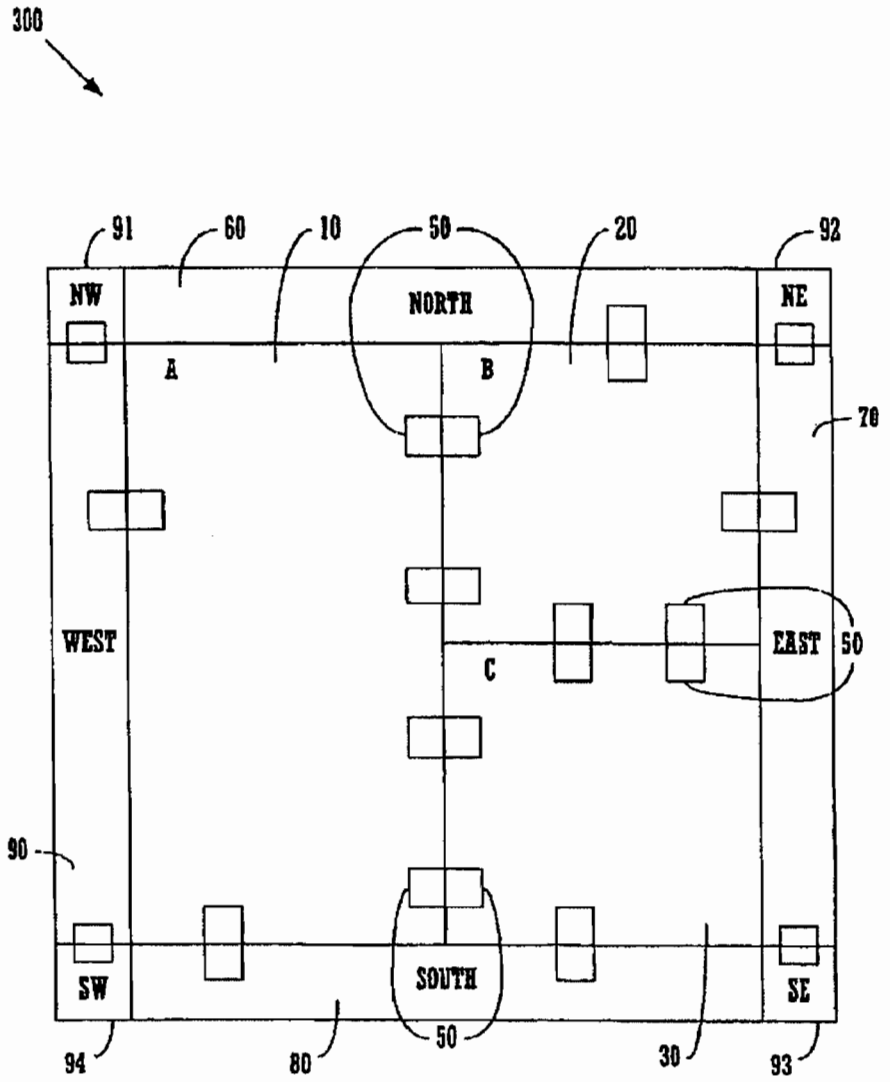

300
91
60
10
50
20
92
NW
NORTH
NE
A
B
70
WEST
EAST
50
C
90
SW
SOUTH
SE
50
94
80
30
93

FIGURE 3

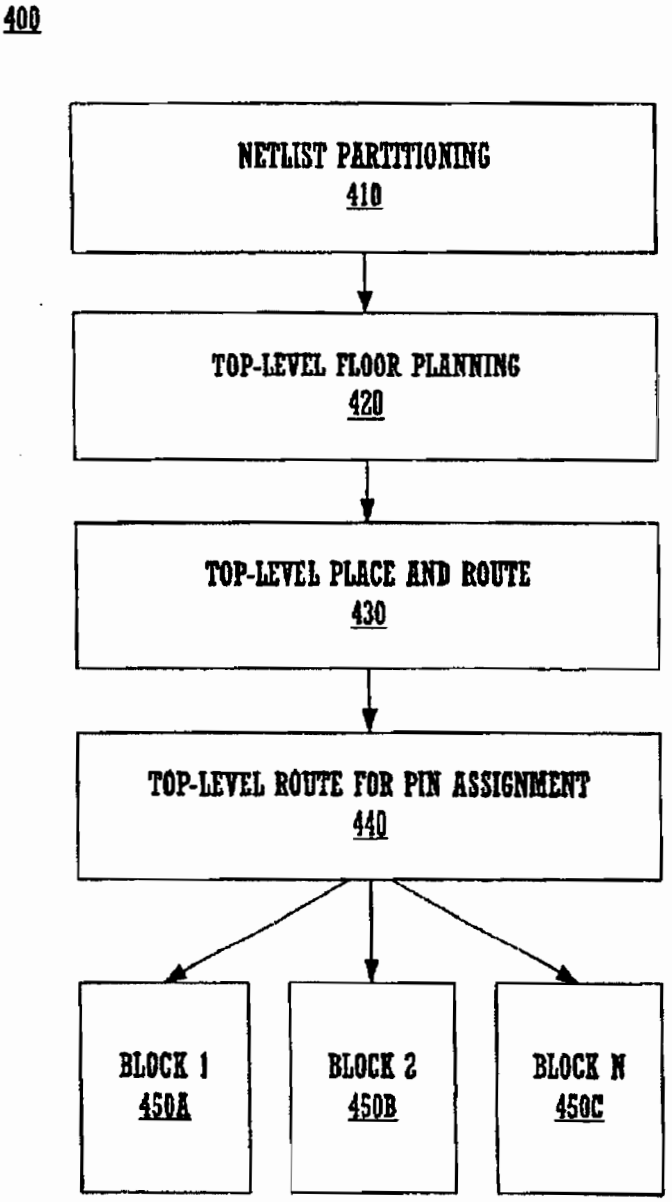

400
NETLIST PARTITIONING
410
TOP-LEVEL FLOOR PLANNING
420
TOP-LEVEL PLACE AND ROUTE
430
TOP-LEVEL ROUTE FOR PIN ASSIGNMENT
440
BLOCK 1
450A
BLOCK 2
450B
BLOCK N
450C

FIGURE 4

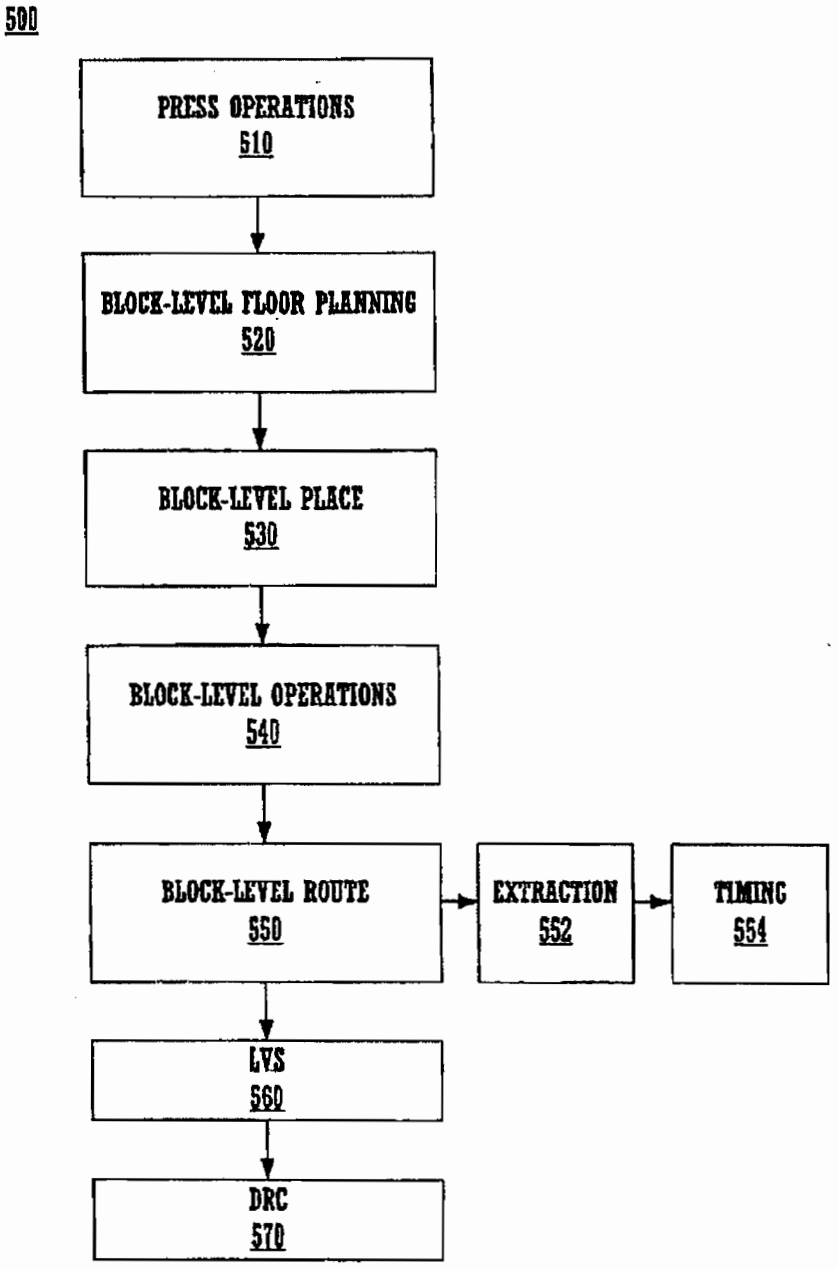

500
PRESS OPERATIONS
510
BLOCK-LEVEL FLOOR PLANNING
520
BLOCK-LEVEL PLACE
530
BLOCK-LEVEL OPERATIONS
540
BLOCK-LEVEL ROUTE
550
EXTRACTION
552
TIMING
554
LVS
560
DRC
570

FIGURE 5

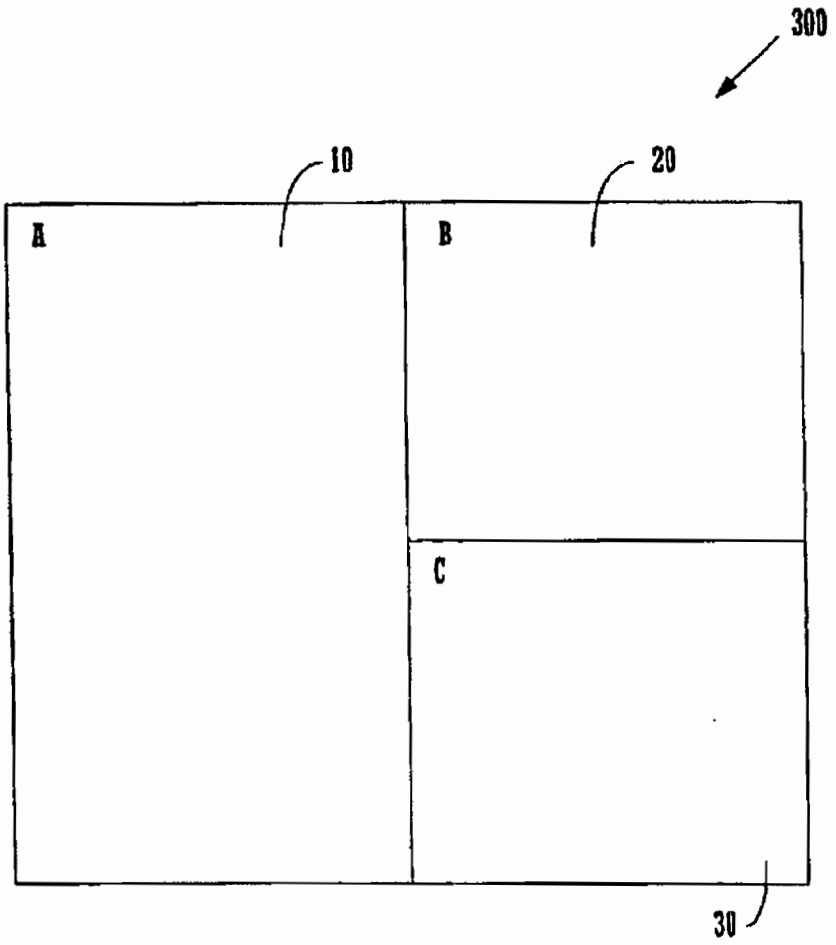


FIGURE 6

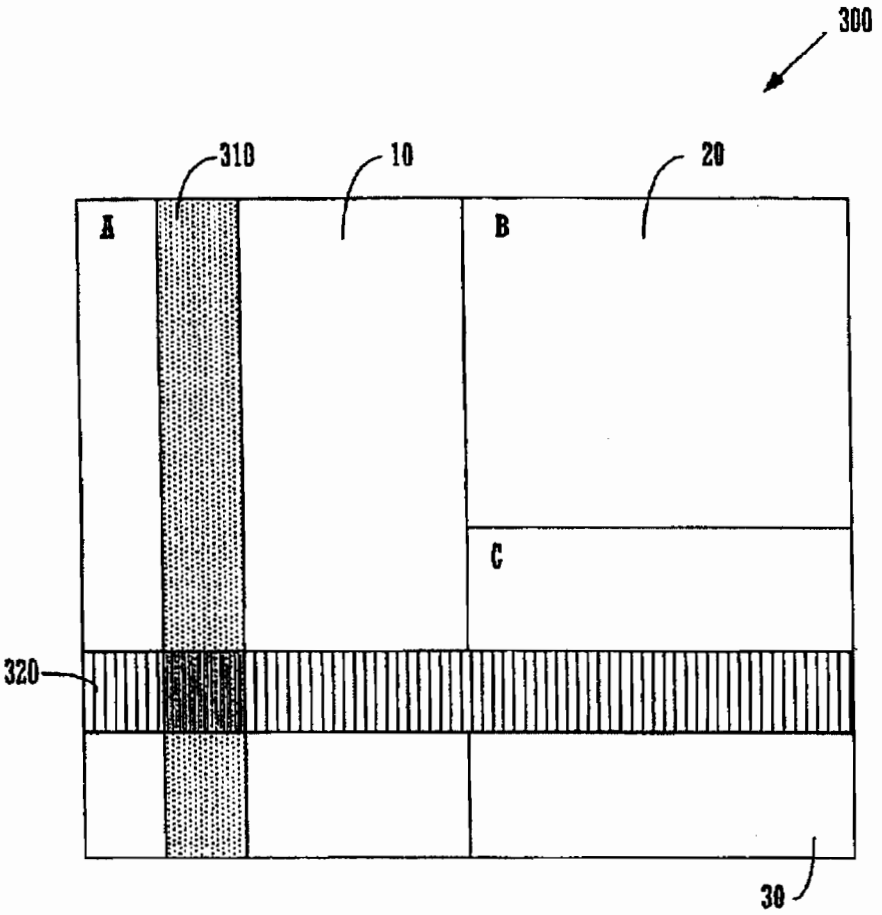


FIGURE 7

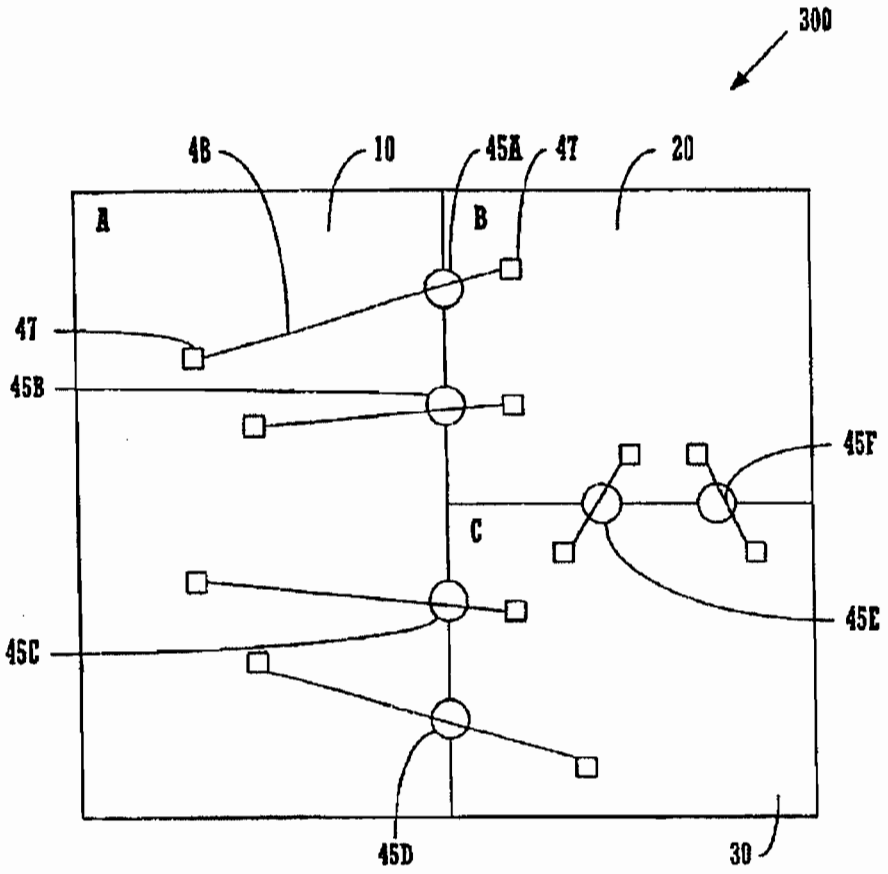


FIGURE 8

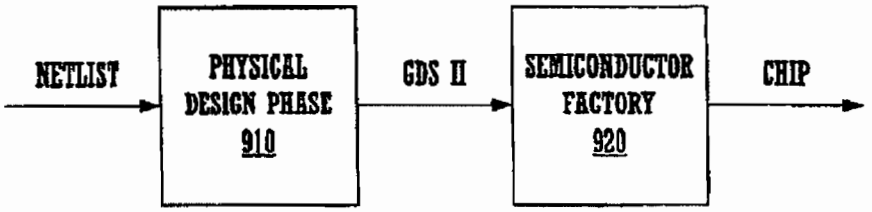


FIGURE 9A
(Prior Art)

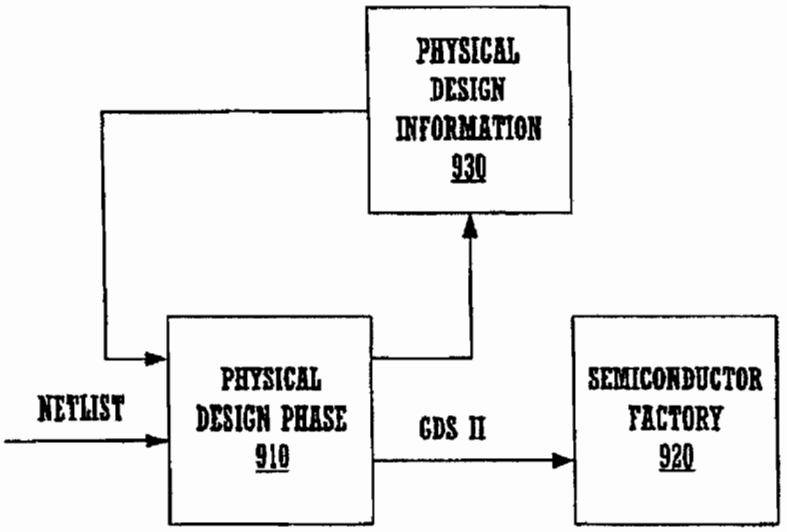

PHYSICAL DESIGN INFORMATION 930
NETLIST
PHYSICAL DESIGN PHASE 910
GDS II
SEMICONDUCTOR FACTORY 920

FIGURE 9B

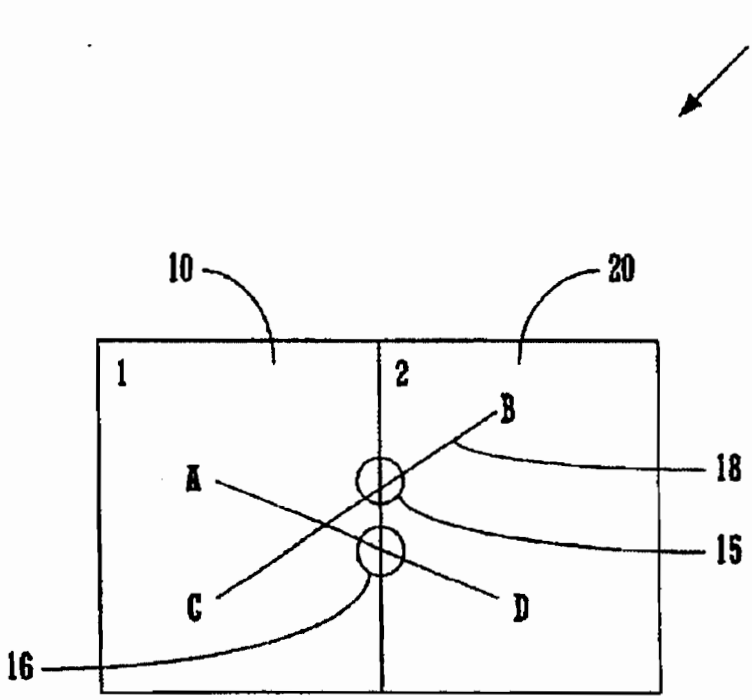


FIGURE 10A

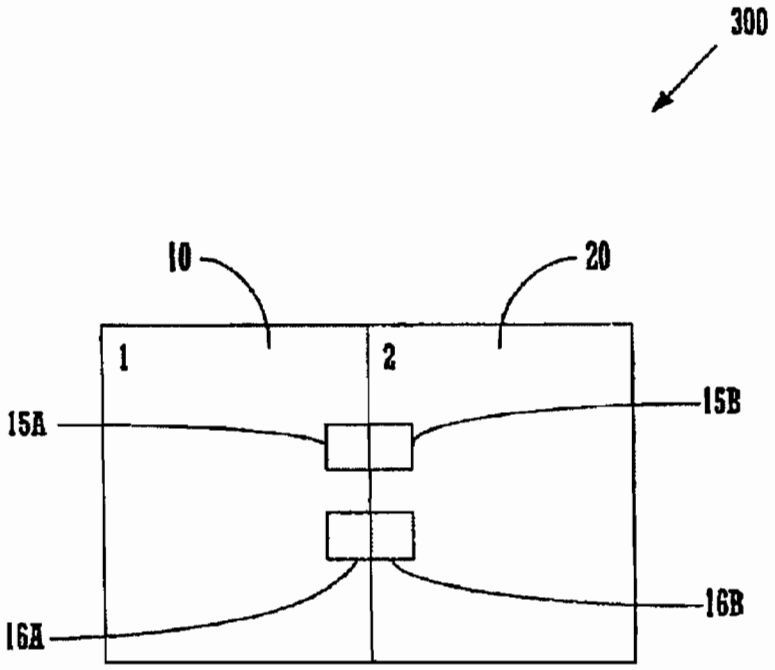


FIGURE 10B

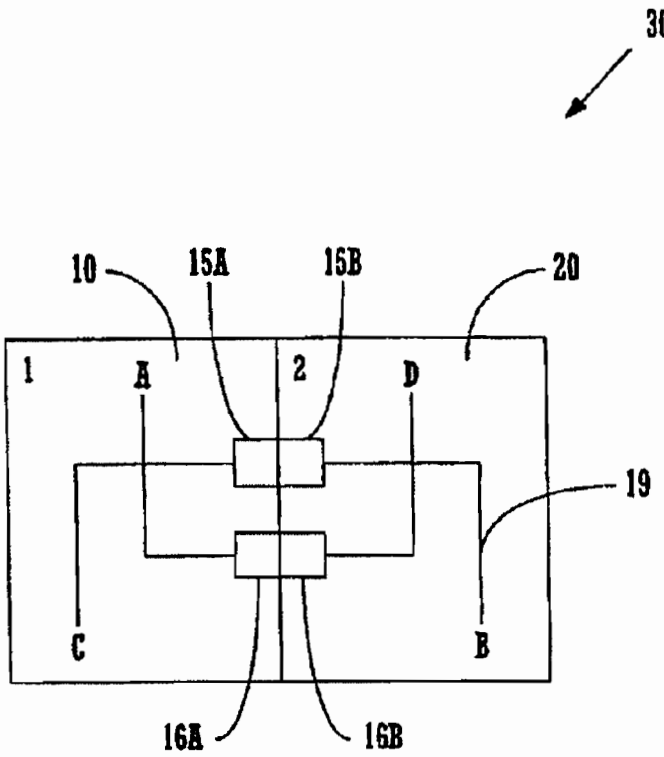


FIGURE 10C

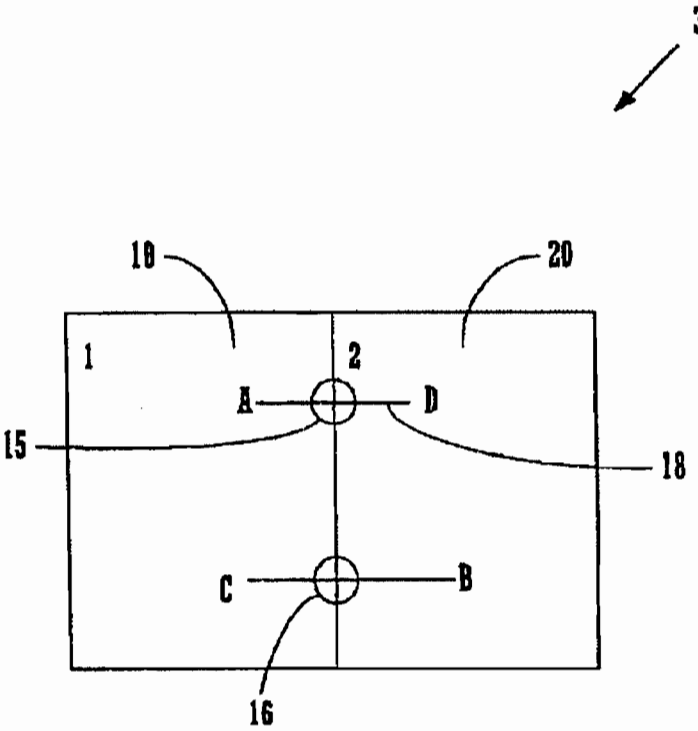


FIGURE 11A

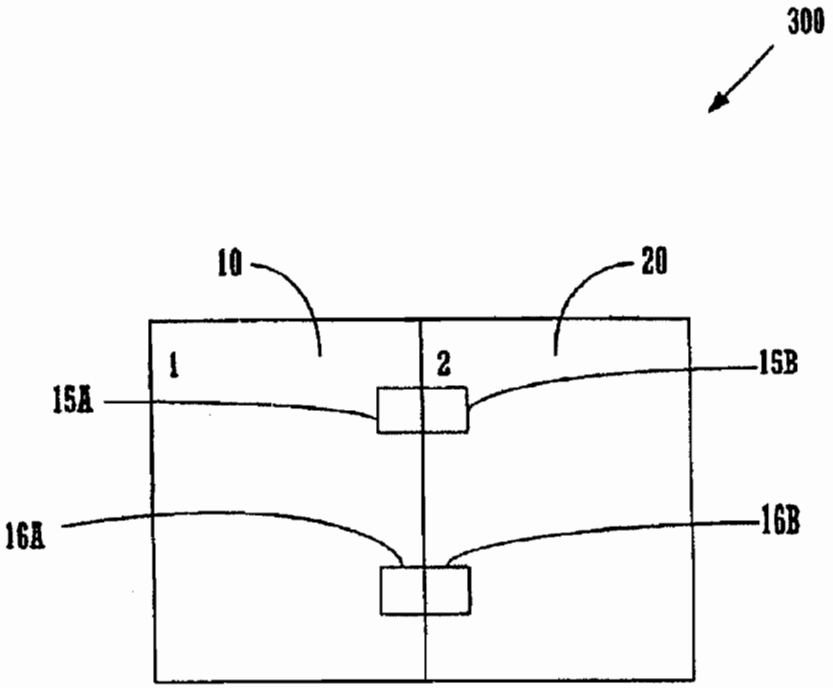


FIGURE 11B

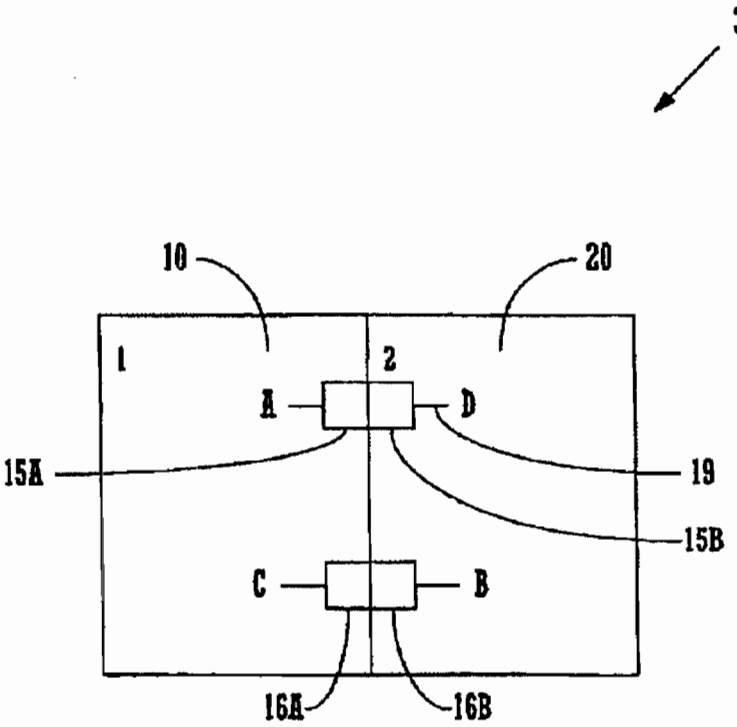


FIGURE 11C

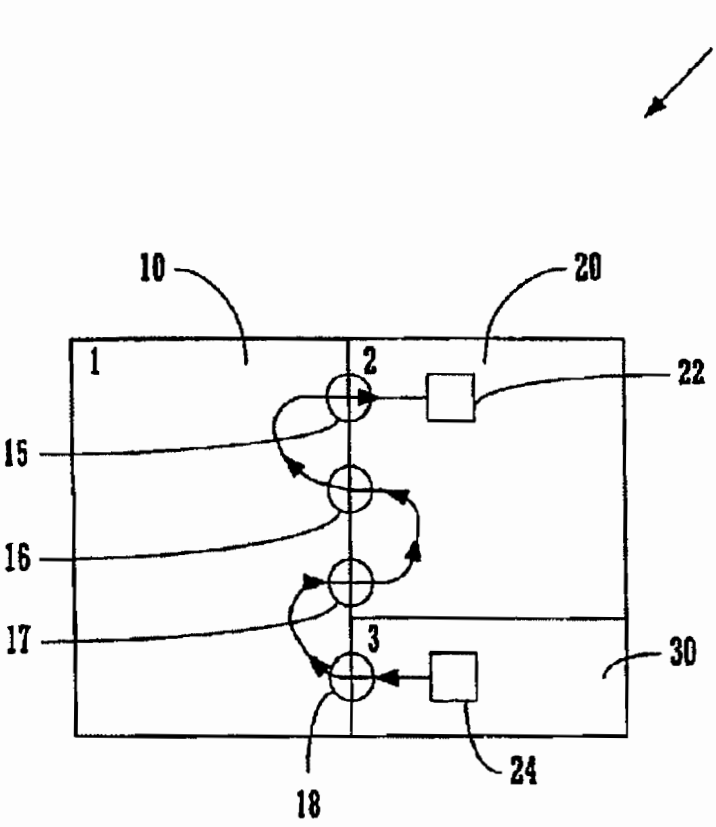


FIGURE 12A

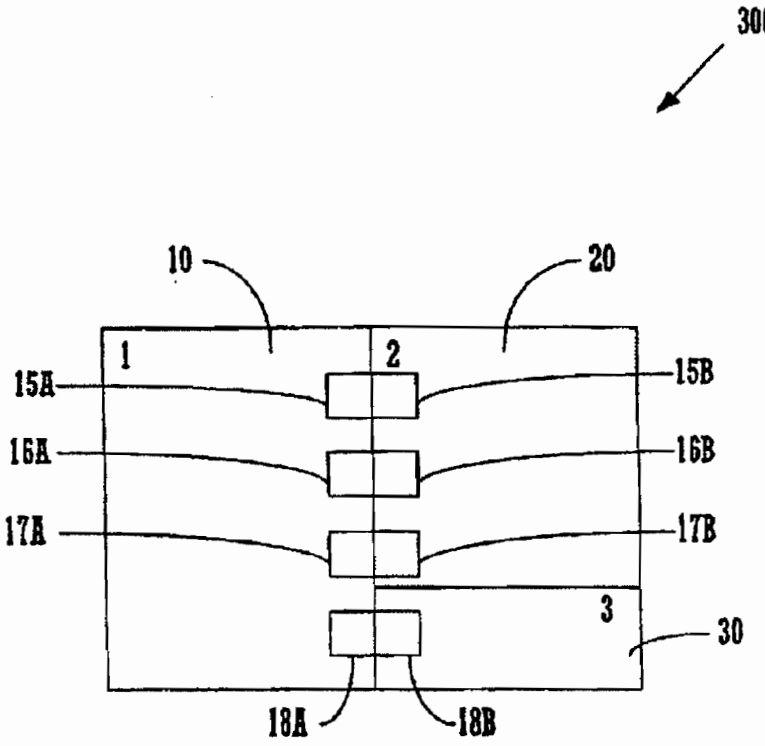


FIGURE 12B

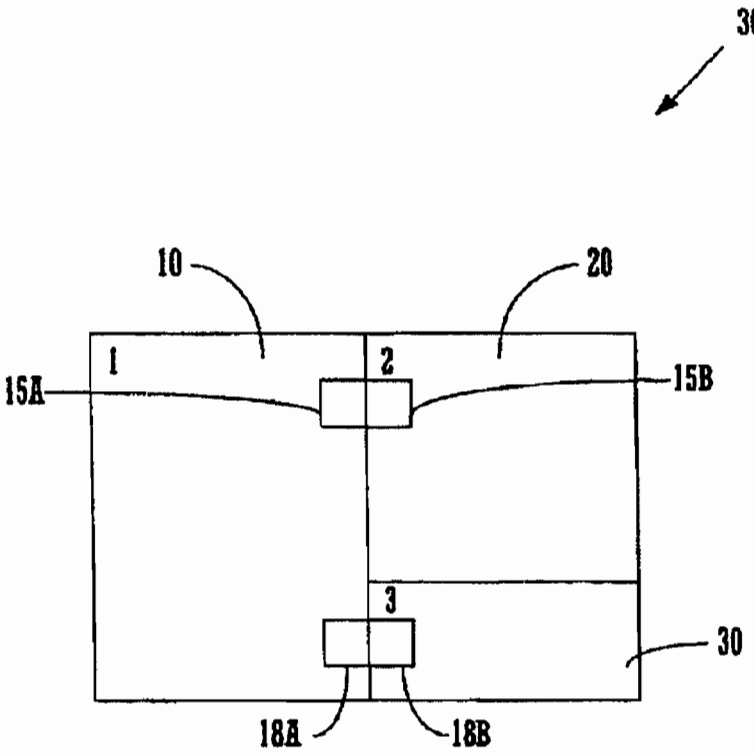


FIGURE 12C

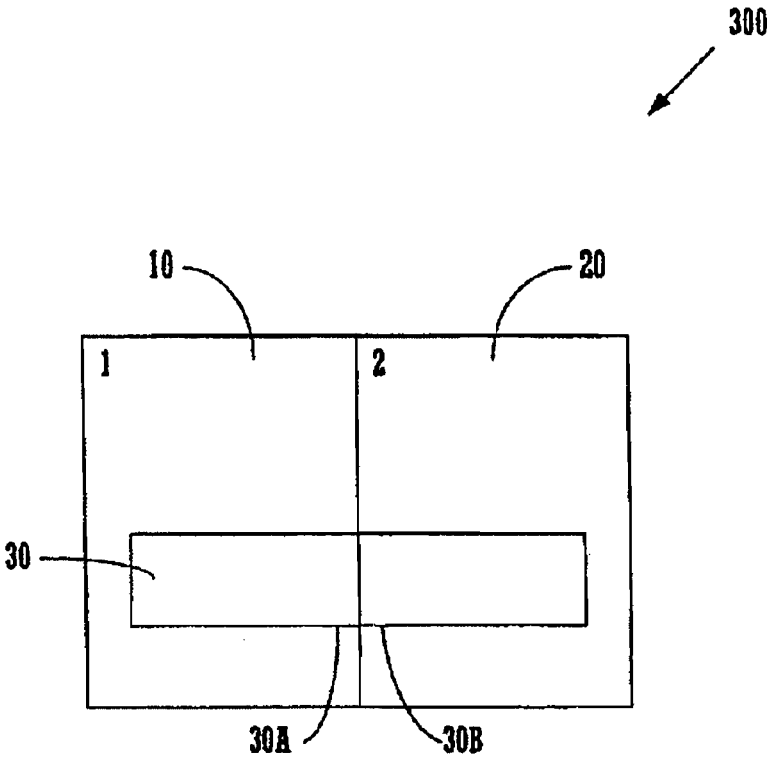


FIGURE 13A

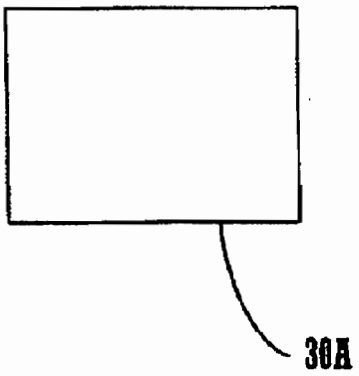


FIGURE 13B

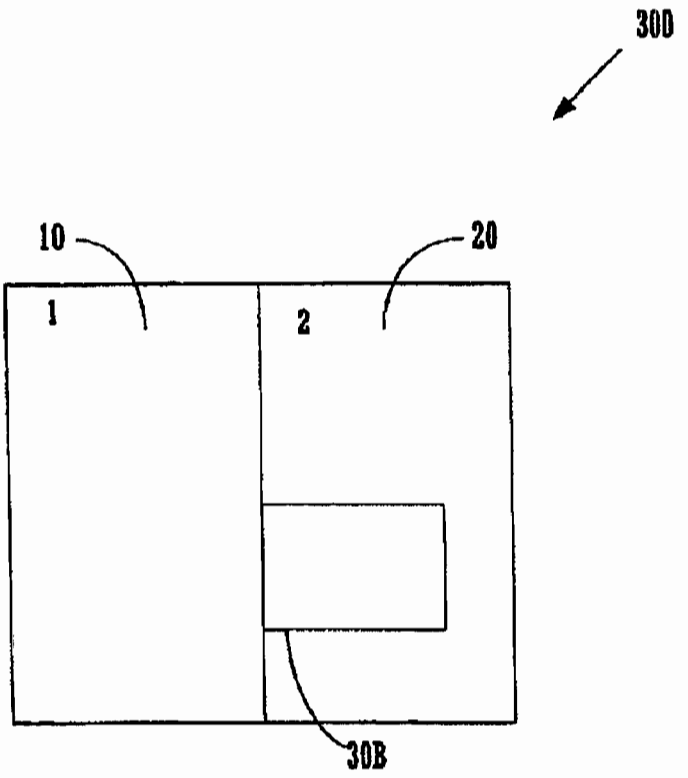


FIGURE 13C

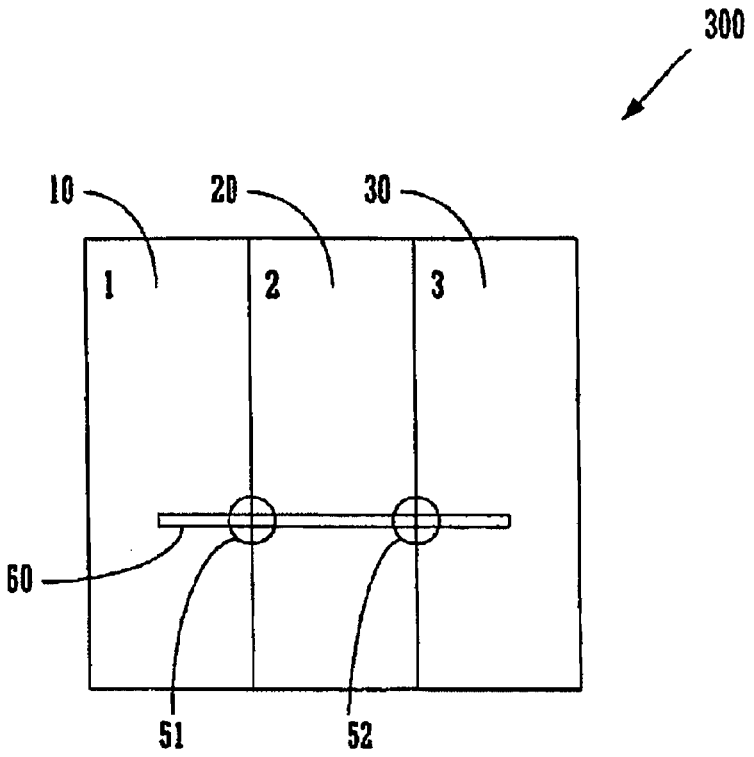


FIGURE 14A

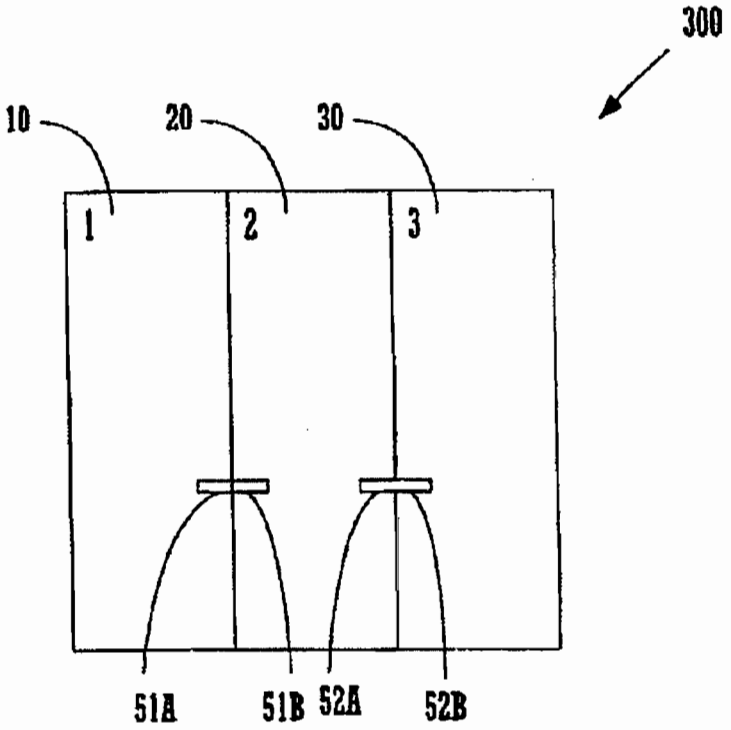


FIGURE 14B

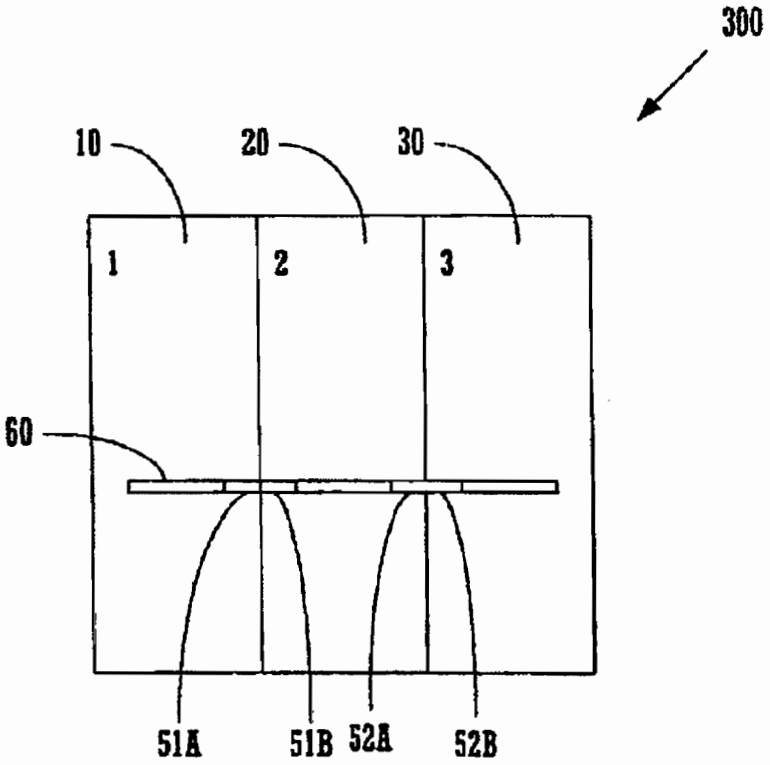


FIGURE 14C

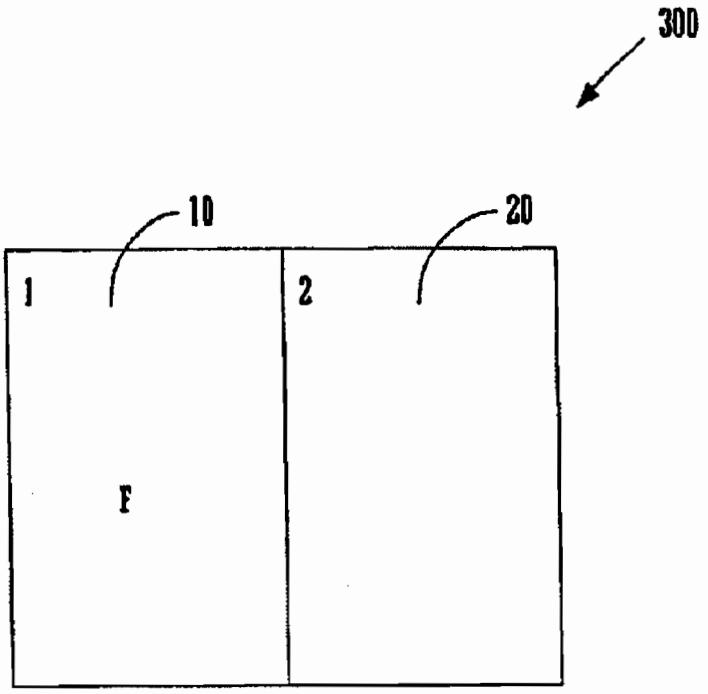


FIGURE 15A

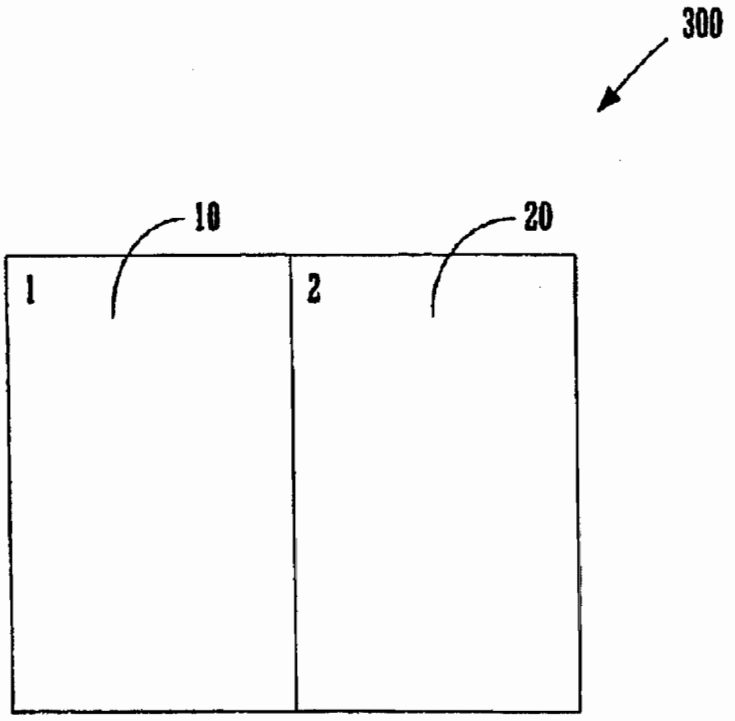


FIGURE 15B

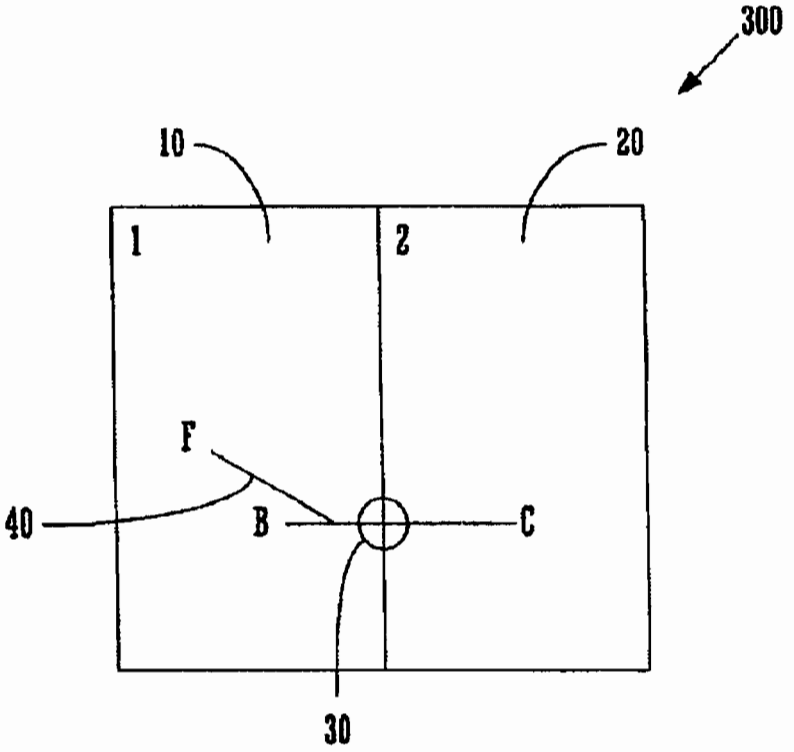


FIGURE 16A

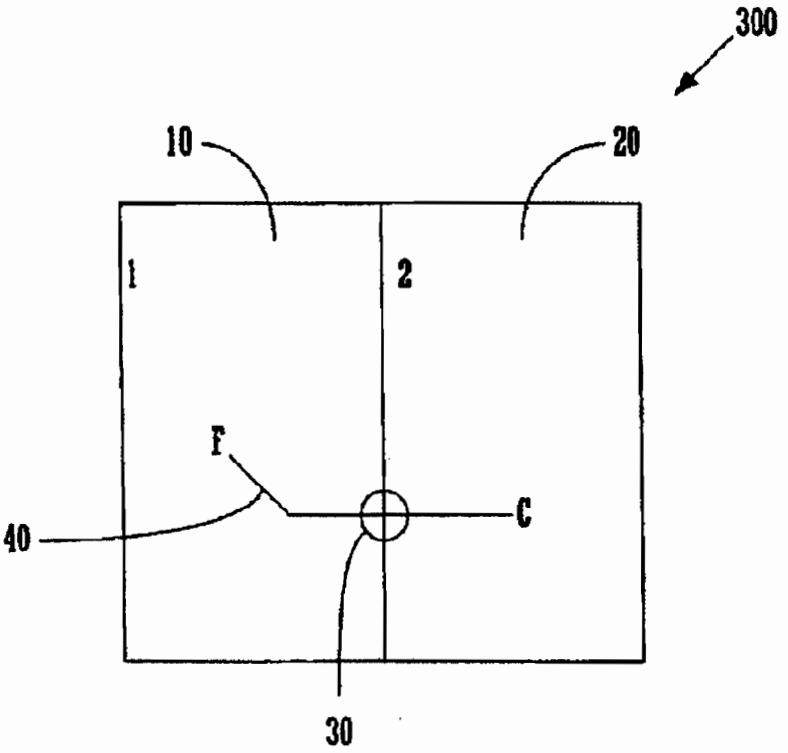


FIGURE 16B

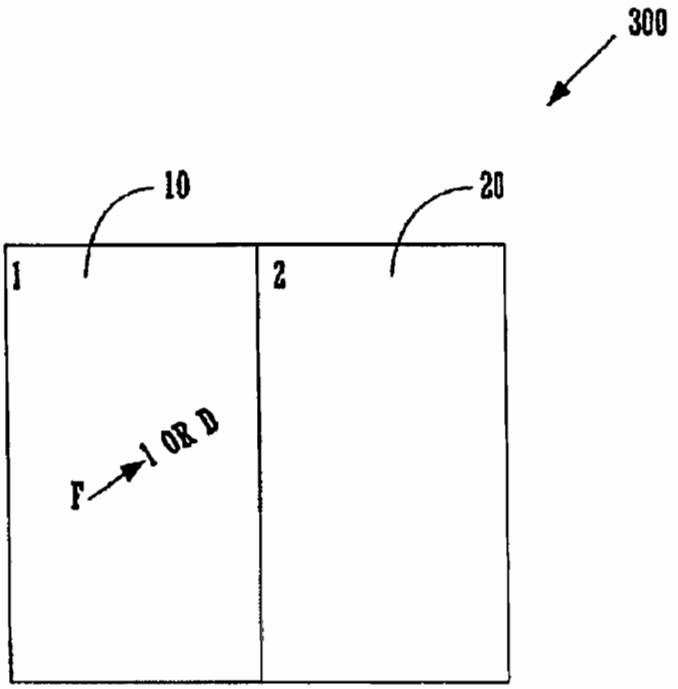


FIGURE 17A

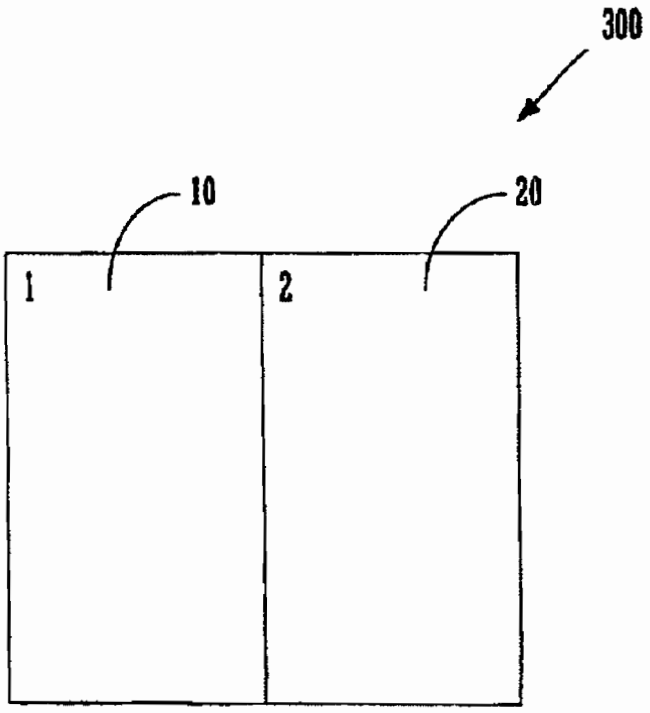


FIGURE 17B

OPTIMIZATION OF ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention generally relates to the field of integrated circuit design. More particularly, the present invention relates to the field of software tools for hierarchical physical design

2. Related Art

The tremendous advances in technology have been fueled by improvements in integrated circuit design. In particular, integrated circuits have become smaller and more complex. Integrated circuit design engineers depend on electronic design automation (EDA) software tools to facilitate the design of integrated circuits

Typically, the integrated circuit design process begins with a specification which describes the functionality of the integrated circuit and may include a variety of constraints. Then, during a logic design phase, the logical implementation of the integrated circuit is determined. Several operations are performed to obtain a logical representation of the integrated circuit. Generally, EDA software tools use register transfer logic (RTL) to represent the integrated circuit. However, additional EDA software tools may be used

After completing the logic design phase, the integrated circuit undergoes a physical design phase. Typically, the output of the logic design phase, is a netlist, which is then used in the physical design phase. Here, EDA software tools layout the integrated circuit to obtain a representation of the physical components in the integrated circuit, whereas the representation indicates the manner in which the integrated circuit will be implemented on a semiconductor chip. A variety of operations are performed on the layout of the integrated circuit

At the end of the physical design phase, the representation of the semiconductor chip (in which the integrated circuit is implemented) is sent to a semiconductor manufacturing plant

Typically, in the physical design phase, EDA software tools implement a flat physical design. For example, the components (standard cells, macrocells, etc.) of the integrated circuit are placed during a placement operation and are routed during a routing operation. However, as the integrated circuit becomes more complex, the EDA software tools struggle to perform the placement operation and the routing operation. In particular, the performance of the EDA software tools degrades since the EDA software tools have to manipulate very large files during the placement operation and the routing operation. Moreover, as the complexity of the integrated circuit increases, the time necessary to complete the physical design phase increases significantly.

Traditional hierarchical physical design has emerged as an alternative to the flat physical design. FIG. 1 illustrates the traditional hierarchical physical design **100**. Here, the components of the integrated circuit are partitioned into a plurality of blocks **10–30**. Each block **10–30** includes a plurality of pins **50**, whereas each pin **50** represents a location where a signal can enter the block **10–30** or a location where a signal can exit the block **10–30**. As illustrated in FIG. **1**, the traditional hierarchical physical design **100** includes a channel **40**. The channel **40** provides space in order to connect the pins **50** of the blocks **10–30** to one another via metal (not shown) or any other wiring material. The traditional hierarchical physical design **100** enables the placement operation and the routing operation (as well as other operations) for the blocks **10–30** to be performed in parallel with EDA software tools, reducing the time period of the physical design phase. Moreover, the performance of the EDA software tools is improved because the file for each block **10–30** is much smaller than the file for the entire integrated circuit of the flat physical design. More importantly, the EDA software tools are better suited to optimize each block **10–30** than to optimize the entire integrated circuit of the flat physical design. However, the traditional hierarchical physical design **100** generates wasted space in the channel **40** and generates wiring problems in the channel **40**, such as congestion and crosstalk. Moreover, the traditional hierarchical physical design **100** places and routes components at a top-level (shown in FIG. **1**) and a block-level (within each block **10–30**), causing inefficiencies and causing problems with EDA software tools which are configured to operate with flat physical designs.

SUMMARY OF THE INVENTION

An abutted-pin hierarchical physical design process is described. The abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design and provides additional advantages and benefits. In particular, the abutted-pin hierarchical physical design does not have channels. Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced significantly.

In the integrated circuit design flow according to an embodiment of the present invention, the physical design phase receives the netlist from the logic design phase. In addition, the physical design phase receives physical design information, whereas the physical design information can be any information about a prior integrated circuit that has undergone the physical design phase. In an embodiment, the physical design information is stored in a database.

In an embodiment of the present invention, the integrated circuit design flow of the present invention is utilized to optimize pin assignment. In an embodiment of the present invention, excess pins formed along a boundary between two blocks are removed.

In an embodiment of the present invention, a software tool that performs a "press" operation preserves the properties associated with a segment of a top-level shape despite the shape operation (e.g., AND) being performed with the block and the top-level shape to obtain the segment.

If the top-level object has the press property, the top-level object retains its location when the top-level object is "pressed" into a block. If the top-level object does not have the press property, the top-level object generally does not retain its location when the top-level object is "pressed" into the block.

If in the top-level netlist, the instantiation of a block includes a port that is unused, (thus, not needed for the top-level routing for pin assignment), a software tool removes the port from the top-level netlist, but the block-level netlist of the block remains unchanged.

Some software tools are not able to represent the relationship that more than one port is coupled to a pin. Hence, a software tool removes one of the ports from the netlist based on some criteria, such as whether a port is an input port or an output port.

If in the top-level netlist, the instantiation of the block includes a port that is tied to either the power line (1) or the

ground line (0) rather to a port of another block, a software tool removes the port from the top-level netlist to avoid routing the port at the top-level. Moreover, the software tool ties the port to either the power line (1) or the ground line (0) in the block-level netlist of the block.

In an embodiment, a software tool performs an unwinding operation which adds to the block-level netlist—of bonding pad blocks—the ports (which were removed earlier by the software tool) that couple to the top-level inputs and to the top-level outputs. Thus, the netlist modified by the physical design phase (e.g., repeater and buffers are added to the netlist) can be compared with the netlist originally received from the logic design phase. In particular, formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification can be performed by software tools.

In an embodiment, each block-level netlist is partitioned into a first netlist and a second netlist. The second netlist and its associated extraction file of each block and the top-level netlist and its associated extraction file are utilized by software tools to perform the timing analysis. This timing analysis can be performed significantly faster than the case where the block-level netlist is not partitioned into the first netlist and the second netlist. In an embodiment, the timing graph resulting from the timing analysis can be analyzed to extract timing constraints (relating to the delay that can be generated by a block) for each block. Hence, if a block is optimized to meet its extracted timing constrains, the block is more likely to meet its timing parameter when the block interacts with the other blocks in the integrated circuit.

These and other advantages of the present invention will no doubt become apparent to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the present invention.

FIG. 1 illustrates the traditional hierarchical physical design 100.

FIG. 2 illustrates an exemplary computer system 200 on which embodiments of the present invention may be practiced.

FIG. 3 illustrates an integrated circuit 300 generated with software tools according to an embodiment of the abutted-pin hierarchical physical design process of the present invention.

FIG. 4 illustrates the abutted-pin hierarchical physical design process 400 according to an embodiment of the present invention.

FIG. 5 illustrates the abutted-pin hierarchical physical design process 500 as performed at the block-level in a particular block (450A–450C of FIG. 4) after step 440 of FIG. 4.

FIG. 6 illustrates the layout of the blocks 10–30 is established.

FIG. 7 illustrates a clock wire 320 and a power wire 310 of the top-level.

FIG. 8 illustrates a top-level route for obtaining the pin assignments for each block 10–30.

FIG. 9A illustrates the integrated circuit design flow of the prior art.

FIG. 9B illustrates the integrated circuit design flow according to an embodiment of the present invention.

FIG. 10A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the prior art (FIG. 9A), showing the top-level routing for pin assignment.

FIG. 10B illustrates the integrated circuit 300 of FIG. 10A at the block-level.

FIG. 10C illustrates the integrated circuit 300 of FIG. 10B at the block-level.

FIG. 11A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the present invention (FIG. 9B), showing the top-level routing for pin assignment.

FIG. 11B illustrates the integrated circuit 300 of FIG. 11A at the block-level.

FIG. 11C illustrates the integrated circuit 300 of FIG. 11B at the block-level.

FIG. 12A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 12B illustrates the integrated circuit 300 of FIG. 12A at the block-level.

FIG. 12C illustrates the integrated circuit 300 of FIG. 12B, showing the removal of excess pins.

FIG. 13A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 30 (e.g., routing metal).

FIG. 13B illustrates the segment 30A of FIG. 13A.

FIG. 13C illustrates the integrated circuit 300 of FIG. 13A in the top-level, showing that the segment 30A has been removed from the top-level netlist and merged into the block-level netlist of block1 10.

FIG. 14A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 60 (e.g., routing metal).

FIG. 14B illustrates the integrated circuit 300 at the block-level.

FIG. 14C illustrates the integrated circuit 300 at the block-level.

FIG. 15A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 15B illustrates that the port F of block1 10 has been removed from the top-level netlist.

FIG. 16A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 16B illustrates that the port B of block1 10 has been removed from the netlist for the top-level routing for pin assignment.

FIG. 17A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 17B illustrates that the port F of block1 10 has been removed from the top-level netlist.

The drawings referred to in this description should not be understood as being drawn to scale except if specifically noted

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made in detail to the preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be recognized by one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Notation and Nomenclature

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. In the present application, a procedure, logic block, process, etc., is conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proved convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, a variety of terms are discussed that refer to the actions and processes of an electronic system or a computer system, or other electronic computing device/system. The computer system or similar electronic computing device manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission, or display devices. The present invention is also well suited to the use of other computer systems such as, for example, optical, mechanical, or quantum computers.

Exemplary Computer System Environment

Aspects of the present invention are discussed in terms of steps executed on a computer system. Although a variety of different computer systems can be used with the present invention, an exemplary computer system **200** is shown in FIG. 2.

With reference to FIG. 2, portions of the present invention are comprised of computer-readable and computer executable instructions which reside, for example, in computer-usable media of an electronic system such as the exemplary computer system. FIG. 2 illustrates an exemplary computer system **200** on which embodiments of the present invention may be practiced. It is appreciated that the computer system **200** of FIG. 2 is exemplary only and that the present invention can operate within a number of different computer systems including general-purpose computer systems and embedded computer systems.

Computer system **200** includes an address/data bus **110** for communicating information, a central processor **101** coupled with bus **110** for processing information and instructions, a volatile memory **102** (e.g., random access memory RAM) coupled with the bus **110** for storing information and instructions for the central processor **101** and a non-volatile memory **103** (e.g., read only memory ROM) coupled with the bus **110** for storing static information and instructions for the processor **101**. Exemplary computer system **200** also includes a data storage device **104** ("disk subsystem") such as a magnetic or optical disk and disk drive coupled with the bus **110** for storing information and instructions. Data storage device **104** can include one or more removable magnetic or optical storage media (e.g., diskettes, tapes) which are computer readable memories. Memory units of computer system **200** include volatile memory **102**, non-volatile memory **103** and data storage device **104**.

Exemplary computer system **200** can further include an optional signal generating device **108** (e.g., a network interface card "NIC") coupled to the bus **110** for interfacing with other computer systems. Also included in exemplary computer system **200** of FIG. 2 is an optional alphanumeric input device **106** including alphanumeric and function keys coupled to the bus **110** for communicating information and command selections to the central processor **101**. Exemplary computer system **200** also includes an optional cursor control or directing device **107** coupled to the bus **110** for communicating user input information and command selections to the central processor **101**. An optional display device **105** can also be coupled to the bus **110** for displaying information to the computer user. Display device **105** may be a liquid crystal device, other flat panel display, cathode ray tube, or other display device suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device **107** allows the user to dynamically signal the two-dimensional movement of a visible symbol (cursor) on a display screen of display device **105**. Many implementations of cursor control device **107** are known in the art including a trackball, mouse, touch pad, joystick or special keys on alphanumeric input device **106** capable of signaling movement of a given direction or manner of displacement. Alternatively, it will be appreciated that a cursor can be directed and/or activated via input from alphanumeric input device **106** using special keys and key sequence commands.

Abutted-pin Hierarchical Physical Design

FIG. 3 illustrates an integrated circuit **300** generated with software tools according to the abutted-pin hierarchical physical design process of the present invention. The abutted-pin hierarchical physical design provides solutions

to the problems of the traditional hierarchical physical design (see FIG 1) and provides additional advantages and benefits In particular, the abutted-pin hierarchical physical design does not have channels. Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced to instantiations of each block 10–30 and 60–94

As illustrated in FIG. 3, the abutted-pin hierarchical physical design 300 includes a plurality of blocks 10–30 and 60–94 The netlist of the integrated circuit 300 is partitioned into the plurality of blocks 10–30 and 60–94 such that each block 10–30 and 60–94 has a block level netlist Blocks 10–30 have the major or core components of the integrated circuit 300 Blocks 60–94 have the bonding pads and other support circuitry of the integrated circuit 300 The blocks 10–30 and 60–94 can be rectangular in shape and can be rectilinear in shape It should be understood that the integrated circuit 300 can have any number of blocks

Each block 10–30 and 60–94 has one or more pins 50, whereas each pin 50 represents a location where a signal can enter the block 10–30 and 60–94 or a location where a signal can exit the block 10–30 and 60–94. The edge or boundary of each block 10–30 and 60–94 rests against the edge or boundary of another block 10–30 and 60–94, such that the pin 50 of one block abuts the pin 50 of another block

Moreover, the top-level components or objects (e g , timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc) are not visible because they have been merged into the blocks 10–30 and 60–94 by a "press" operation performed by a software tool. First, the top-level objects (e g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc) are placed and routed at the top-level (the top-level is shown in FIG. 3) In the "press" operation, the top-level objects (e g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc) that are within the boundary of a block 10–30 and 60–94 are removed from the top-level netlist and merged into the block-level netlist of that block 10–30 and 60–94 Hence, the abutted-pin hierarchical physical design 300 can be optimized by separately optimizing the individual blocks 10–30 and 60–94 Thus, the software tools can generate (e g., perform placement, routing, timing, verification, etc) and optimize the individual blocks 10–30 and 60–94 in parallel Moreover, a bug within an individual block 10–30 and 60–94 can be corrected by returning that individual block to the logic design phase, while the other blocks continue to undergo the physical design phase

FIG. 4 illustrates the abutted-pin hierarchical physical design process 400 according to an embodiment of the present invention At 410, a software tool receives the netlist of the integrated circuit from the logic design phase, as described above. The netlist is partitioned into a plurality of blocks, each block having a block-level netlist In an embodiment, the partitioning of the netlist focuses on reducing the number of ports or terminals of a block that need to couple to the ports or terminals of other blocks

At 420, a software tool performs top-level floor planning. Here, the layout of each block is determined At the end of the top-level floor planning, the top-level for an integrated circuit 300 (as shown in FIG. 6) is generated As illustrated in FIG. 6, the layout of the blocks 10–30 is established In FIG 6, the bonding pads 60–94 (of FIG 3) have been omitted

At 430, software tools perform top-level placement and routing for the top-level objects (e g , timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc). FIG 7 illustrates a clock wire 320 and a power wire 310 of the top-level The clock wire 320 is routed over BlockA 10 and BlockC 30 The power wire 310 is routed over BlockA 10. It should be understood that any number of additional top-level objects can be placed and routed at the top-level.

At 440, a software tool performs a top-level route for obtaining the pin assignments for each block 10–30, as illustrated in FIG 8 Since each block 10–30 has one or more ports or terminals 47 that needs to couple to a port or terminal of another block 10–30, the pins for each block 10–30 have to be defined Initially, the ports 47 of each block 10–30 are placed in a general random location within each block at the top-level since the actual location of the port 47 is not known until a placement operation is performed at the block-level As illustrated in FIG 8, the location 45A–45F where a routing wire 48 crosses a boundary between two blocks is defined as a pin for each of the blocks 10–30, facilitating creation of pins that are abutted In an embodiment, a software tool creates each pin to have a width that is equivalent to the width of the routing wire 48 at the boundary between the two blocks The pins 50 are illustrated in FIG 3

At 450A–450C, the abutted-pin hierarchical physical design process 400 enables software tools to generate and to optimize each block 10–30 in parallel at the block-level

FIG 5 illustrates the abutted-pin hierarchical physical design process 500 as performed at the block-level in a particular block (450A–450C of FIG 4) after step 440 of FIG 4.

At 510, a software tool performs press operations. The top-level objects illustrated in FIG 7 (e g , a clock wire 320 and a power wire 310) and which are located within the boundary of a particular block, are pressed into the particular block. In particular, the top-level objects that are within the boundary of a particular block are removed from the top-level netlist and merged into the block-level netlist of that particular block Moreover, the pins for the particular block are generated based on the location where the routing wire crosses the boundary between two blocks, as illustrated in FIG 8 and FIG. 3

At 520, a software tool performs block-level floor planning for the particular block At 530, a software tool performs a block-level placement operation for the particular block At 540, software tools perform a variety of block-level operations to optimize the particular block. Additionally, at 550, a block-level route is performed for the particular block by a software tool At 552 and 554, software tools perform a block-level extraction operation for determining capacitance and resistance at the nodes and perform block-level timing analysis operations for the particular block

At 560 and 570, a variety of software tools perform a number of verification operations such as formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification.

FIG 9A illustrates the integrated circuit design flow of the prior art. As illustrated in FIG 9A, the physical design phase 910 receives the netlist from the logic design phase (not shown) The physical design phase 910 generates the physical design for the integrated circuit and outputs a GDS II file. The GDS II file is received by the semiconductor factory 920. The integrated circuit is fabricated by the semiconductor factory 920 on a semiconductor chip.

FIG. 9B illustrates the integrated circuit design flow according to an embodiment of the present invention As

illustrated in FIG. 9B, the physical design phase 910 receives the netlist from the logic design phase (not shown). In addition, the physical design phase 910 receives physical design information 930, whereas the physical design information 930 can be any information about a prior integrated circuit that has undergone the physical design phase 910. In an embodiment, the physical design information 930 is stored in a database. For example, the physical design information 930 can be pin assignments of the prior integrated circuit, optimal clock distribution tree of the prior integrated circuit, parasitic extraction data of the prior integrated circuit, locations of obstructions such as a RAM of the prior integrated circuit, identification of congested blocks of the prior integrated circuit, metal resources for the blocks of the prior integrated circuit, or any other information which can facilitate optimizing the current integrated circuit. Thus, the software tools of the physical design phase 910 can customize the current integrated circuit to avoid the problems of the prior integrated circuit and to realize the benefits of the prior integrated circuit.

In the physical design phase 910, decisions made at the top-level with respect to the top-level objects, significantly influence the creation of problems at the block-level and the optimization operations at the block-level. By using physical design information 930 (concerning the block-level of the prior integrated circuit) at the top-level of the current integrated circuit, the decisions made at the top-level with respect to the top-level objects of the current integrated circuit will be able to reduce the problems present in the prior integrated circuit and will be able to generate solutions to overcome the problems present in the prior integrated circuit, improving the optimization of the abutted-pin hierarchical physical design process of the present invention. Thus, if the physical design information 930 has information about several prior integrated circuits, the current integrated circuit is more likely to be optimized.

In addition, the physical design phase 910 generates the physical design for the integrated circuit and outputs a GDS II file. Moreover, the physical design phase 910 stores physical design information 930 of the current integrated circuit to be used in the physical design phase 910 of a future integrated circuit. The GDS II file is received by the semiconductor factory 920. The integrated circuit is fabricated by the semiconductor factory 920 on a semiconductor chip.

FIG. 10A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the prior art (FIG. 9A), showing the top-level routing for pin assignment. The port C of block1 10 is routed to port B of block2 20. The port A of block1 10 is routed to port D of block2 20. This top-level routing has been performed after ports A–D where placed in a generally random location within each block 10–20 at the top-level since the actual locations of the ports A–D are not known until a placement operation is performed at the block-level. Here, the software tools at the top-level do not have access to the physical design information of a prior integrated circuit. The locations 15 and 16 are where the routing metal 18 crosses the boundary between two blocks 10 and 20.

FIG. 10B illustrates the integrated circuit 300 of FIG. 10A. At the block-level, the pins 15A and 16A were formed for block1 10. At the block-level, the pins 15B and 16B were formed for block2 20, whereas pin 15A abuts pin 15B and pin 16A abuts pin 16B. The pins 15A and 15B were formed at location 15 of FIG. 10A. The pins 16A and 16B were formed at location 16 of FIG. 10A.

FIG. 10C illustrates the integrated circuit 300 of FIG. 10B at the block-level. As illustrated in FIG. 10C, the block-level placement operation for block1 10 placed the ports A and C at locations that are different from the locations used to generate the pin assignments in FIG. 10A. In addition, the block-level placement operation for block2 20 placed the ports B and D at locations that are different from the locations used to generate the pin assignments in FIG. 10A. Hence, the block-level routing operations for blocks 10 and 20 generated an inefficient amount of routing wire 19 to couple the ports to the pins in each block. In sum, the pin assignment affects the optimization of the routing wire 19.

FIG. 11A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the present invention (FIG. 9B), showing the top-level routing for pin assignment. The port C of block1 10 is routed to port B of block2 20. The port A of block1 10 is routed to port D of block2 20. This top-level routing has been performed after each port A–D where placed in a particular location within each block 10–20 at the top-level, whereas the particular location was based on using the physical design information associated with the prior integrated circuit (FIGS. 10A–10C). Here, the software tools at the top-level have access to the physical design information of the prior integrated circuit (FIGS. 10A–10C). The locations 15 and 16 are where the routing metal 18 crosses the boundary between two blocks 10 and 20.

FIG. 11B illustrates the integrated circuit 300 of FIG. 11A at the block-level. At the block-level, the pins 15A and 16A were formed for block1 10. At the block-level, the pins 15B and 16B were formed for block2 20, whereas pin 15A abuts pin 15B and pin 16A abuts pin 16B. The pins 15A and 15B were formed at location 15 of FIG. 1A. The pins 16A and 16B were formed at location 16 of FIG. 11A. Here, the pins 15A and 15B are associated with ports A and D, unlike FIG. 10B where pins 15A and 15B were associated with ports C and B. Moreover, the pins 16A and 16B of FIG. 11B are associated with ports C and B, unlike FIG. 10B where pins 16A and 16B were associated with ports A and D.

FIG. 11C illustrates the integrated circuit 300 of FIG. 11B at the block-level. As illustrated in FIG. 11C, the block-level placement operation for block1 10 placed the ports A and C at locations that are different from the locations used to generate the pin assignments in FIG. 11A. In addition, the block-level placement operation for block2 20 placed the ports B and D at locations that are different from the locations used to generate the pin assignments in FIG. 11A. However, the difference in the location of the ports between FIG. 11A and FIG. 11C is less than the difference in the location of the ports between FIG. 10A and FIG. 10C. Hence, the block-level routing operations for blocks 10 and 20 generated a more efficient amount of routing wire 19 to couple the ports to the pins in each block, compared to FIG. 10C. In sum, the pin assignments generated with the use of the physical design information of the prior integrated circuit (FIGS. 10A–10C) were more optimal than the pin assignments generated without the use of the physical design information of the prior integrated circuit (FIGS. 10A–10C).

FIG. 12A illustrates an integrated circuit 300 based on the abutted-pin a hierarchical physical design process of the present invention, showing the top-level routing for pin assignment. In the course of routing source port 24 of block3 30 to destination port 22 of block2 20, the software tool that performs the top-level routing for pin assignment crosses the boundary between block1 10 and block2 20 at locations 15,16, and 17, whereas the locations 15,16, and 17 will be defined as pins. The software tool is concerned with routing a path between the source port 24 and the destination port

22, but is not concerned about the number of times the path crosses the boundary between the same blocks

FIG 12B illustrates the integrated circuit 300 of FIG. 12A at the block-level The pins 15A–15B, 16A–16B, and 17A–17B are formed between block1 10 and block2 20. The pins 18A–18B are formed between block1 10 and block3 30 The presence of pins 16A–16B and 17A–17B causes additional routing metal to be added to block1 10 and block2 20 so that pins 15A, 16A, and 17A can be coupled within block1 10 and so that pins 15B, 16B, and 17B can be coupled within block2 20. Hence, one pair of pins (15A–15B or 16A–16B or 17A–17B) is sufficient

FIG. 12C illustrates the integrated circuit 300 of FIG 12B, showing the removal of excess pins As illustrated in FIG 120, excess pins 16A–16B and 17A–17B were removed from block1 10 and block2 20 This removal is based on a plurality of criteria, such as the current flow direction between the source port 24 and the destination port 22, the location of the excess pins relative to the source port 24 and the destination port 22, or any other criteria. Here, the criteria kept pins 15A–15B but deleted pins 16A–16B and 17A–17B

FIG 13A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 30 (e.g., routing metal). As described above, a software tool performs a press operation so that the portion of the top-level object 30 which is within the boundary of a particular block 10–20 is moved from the top-level netlist to the block-level netlist of the particular block 10–20. In particular, the segment 30A is pressed into block1 10 while the segment 30B is pressed into block2 20. In an embodiment, the shape operations of a database are utilized in performing the press operation. In FIG. 13A, an AND operation would be performed with block1 10 and the shape 30 to obtain the segment 30A (FIG. 13B) Typically, the routing metal 30 includes a plurality of properties that are stored in a database These properties identify the routing metal 30 and describe the function of the routing metal 30 However, in the shape operations (e.g , AND) of the prior art, the shape operation returns the segment 30A (FIG 13B) without its properties Thus, these properties have to be reconstructed.

In the present invention, the software tool t hat performs the press operation preserves the properties associated with segment 30A of the routing metal 30 despite the shape operation (e.g., AND) performed with block1 10 and the shape 30 to obtain the segment 30A (FIG. 13B).

FIG. 13C illustrates the integrated circuit 300 of FIG. 13A in the top-level, showing that the segment 30A has been removed from the top-level netlist and merged into the block-level netlist of block1 10 Moreover, the properties associated with segment 30A at the top level are transferred to the segment 30A at the block-level

FIG. 14A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 60 (e.g , routing metal) As illustrated in FIG 14A, the top-level object 60 is routed through block1 10, block2 20, and block3 30. The locations 51–52 indicate top-level object 60 crosses a boundary between two blocks In an embodiment, a press property is added to the properties of the top-level object 60 stored in a database If the top-level object 60 has the press property, the top-level object 60 retains its location when the top-level object 60 is pressed into block1 10, block 20, and block3 30, as illustrated in the block-level view of the integrated circuit 300 in FIG 14C If the top-level object 60 does not have the press property, the top-level object 60 generally does not retain its location when the top-level object 60 is pressed into block1 10, block2 20, and block3 30, as illustrated in the block-level view of the integrated circuit 300 in FIG 14B For example, top-level objects such as power and ground have the press property. As illustrated in FIG 14B, the pins 51A–51B and 52A–52B are defined. However, the software tool is not constrained to placing the top-level object 60 in the block-level exactly as it was placed at the top-level. Moreover, the top-level object is placed in the block-level of block1 10, block2 20, and block3 30 according to the separate placement and routing requirements of block1 10, block2 20, and block3 30. –FIG 15A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment. As illustrated in FIG. 15A, in the top-level netlist, the instantiation of block1 10 includes port F that is unused, thus, not needed for the top-level outing for pin assignment Hence, a software tool removes port F from the top-level netlist, but the block-level netlist of block1 10 remains unchanged In an embodiment, the software tool that performs the press operation removes the port F FIG. 15B illustrates that the port F of block1 10 has been removed from the top-level netlist

FIG 16A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment As illustrated in FIG 16A, port F and port B of block1 10 are coupled to port C of block2 20 with a routing metal 40. However, at location 30 the routing metal 40 crosses the boundary between block1 10 and block2 20. If a pin is formed within block1 10 at location 30, the pin would be coupled to port F and to port B However, some software tools are not able to represent this relationship (i e., more than one port coupled to a pin) Hence, a software tool removes one of the ports (port F or port B) from the netlist based on some criteria, such as whether a port is an input port or an output port FIG 16B illustrates that the port B of block1 10 has been removed from the netlist for the top-level routing for pin assignment

FIG 17A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment. As illustrated in FIG. 17A, in the top-level netlist, the instantiation of block1 10 includes a port F that is tied to either the power line (1) or the ground line (0) rather to a port of another block Hence, a software tool removes port F from the top-level netlist to avoid routing the port F at the top-level Moreover, the software tool ties the port F to either power line (1) or the ground line (0) in the block-level netlist of block1 10 FIG 17B illustrates that the port F of block1 10 has been removed from the top-level netlist

As illustrated in FIG 3, the integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention includes a North bond pad block 60, an East bond pad block 70, a South bond pad block 80, and a West bond pad block 90, each having bond pad cells The top-level netlist of the integrated circuit 300 includes one or more top-level inputs for receiving external signals and one or more top-level outputs for transmitting signals off the chip. The top-level inputs and the top-level outputs are coupled to bond pad cells Typically, software tools which perform a routing operation are configured to not perform the routing operation if the netlist includes bond pad cells

Since the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** have bond pad cells in the block-level netlist, the software tools refuse to perform the routing operation in these blocks, preventing pins to be formed on the boundary between these blocks and the blocks **10–30** (the core blocks)

In the present invention, the bond pad cells are marked as macrocells rather than bond pad cells, allowing pins to be formed on the boundary between these blocks **60**, **70**, **80**, and **90** and the blocks **10–30** (the core blocks)

Typically, the block-level netlist of the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** include nets to the top-level inputs and nets to the top-level outputs. Generally, the block-level netlist of the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** include nets to the bond pad cells

In an embodiment of the present invention, a software tool removes the nets to the top-level inputs and nets to the top-level outputs so that the physical design of the integrated circuit can be accomplished as described above. In an embodiment, the software tool removes in the block-level netlist the ports that couple to the top-level inputs and to the top-level outputs. Moreover, the software tool adds a property to the nets to the bond pad cells to indicate that these nets are suppose to couple to the top-level inputs and to the top-level outputs, facilitating an unwinding operation to re-establish at the block-level netlist the nets to the top-level inputs and nets to the top-level outputs that were removed earlier The un winding operation adds to the block-level netlist the ports (which were removed earlier) that couple to the top-level inputs and to the top-level outputs Thus, the netlist modified by the physical design phase (e g., repeater and buffers are added to the netlist) can be compared with the netlist originally received from the logic design phase In particular, formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification can be performed by software tools.

A challenge with implementing an integrated circuit based on the abutted-pin hierarchical physical design process of the present invention involves analyzing the timing of signal paths that traverse more than one block The timing of these global paths is difficult to analyzed compared to analyzing the timing of local paths, whereas local paths are signal paths that do not leave a block One method of analyzing the timing of these global paths involves partitioning the block-level netlist of each block into a first netlist and a second netlist The first netlist includes nets which start at a register (or flip-flop) and end at a register (or flip-flop) within the block, whereas each branch of the net also starts at a register (or flip-flop) and ends at a register (or flip-flop) within the block The second netlist includes nets which are coupled to a pin of the block. Generally, the first netlist is ¾ of the initial block-level netlist while the second netlist is ¼ of the initial block-level netlist. If the second netlist ratio is greater than ¼, this indicates inefficient partitioning of the blocks

Once the first netlist and the second netlist are obtain, an extraction operation to obtain parasitic resistance and capacitance is performed on the second netlist of each block In an embodiment, the partitioning of the block-level netlist and the extraction operation in each block are performed in parallel Moreover, an extraction operation is performed on the top-level netlist In an embodiment, a software tool replaces the abutted pins of the top-level netlist with zero ohm resistors.

Some software tools utilized to perform the timing analysis are unable to operate on netlists having nets that are coupled to multiple pins of a block In an embodiment of the present invention, these netlist are transformed by using "assign statements" to assign different names to the nets that are coupled to multiple pins of a block. Hence, each different named net can be coupled to a separate pin of the block

In an embodiment, the second netlist and its associated extraction file of each block and the top-level netlist and its associated extraction file are utilized by software tools to perform the timing analysis This timing analysis can be performed significantly faster than the case where the block-level netlist is not partitioned into the first netlist and the second netlist In an embodiment, the timing graph resulting from the timing analysis can be analyzed to extract timing constraints (relating to the delay that can be generated by a block) for each block. Hence, if a block is optimized to meet its extracted timing constrains, the block is more likely to meet its timing parameter when the block interacts with the other blocks in the integrated circuit.

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents

What is claimed is:

1. A method of improving a physical design of a current integrated circuit, comprising the steps of:

a) receiving a netlist of said current integrated circuit;

b) receiving physical design information from a prior integrated circuit; and

c) generating said physical design based on said netlist and said physical design information

2 A method as recited in claim 1 wherein said physical design information includes pin assignments of blocks of said prior integrated circuit

3 A method as recited in claim 1 wherein said physical design information includes optimal clock distribution tree of said prior integrated circuit

4 A method as recited in claim 1 wherein said physical design information includes parasitic extraction data of said prior integrated circuit

5 A method as recited in claim 1 wherein said physical design information includes identification of congested blocks of said prior integrated circuit

6. A method as recited in claim 1 wherein said physical design information includes metal resources of said prior integrated circuit

7. A method as recited in claim 1 wherein said physical design information includes information which facilitates optimizing said current integrated circuit

8 A method as recited in claim 1 wherein said step c) includes:

generating a top-level physical design of said current integrated circuit using said netlist and said physical design information including block-level physical design information of said prior integrated circuit.

9 A method as recited in claim 1 wherein said step c) includes:

generating a block-level physical design of said current integrated circuit using said netlist and said physical design information including block-level physical design information of said prior integrated circuit.

10 A method as recited in claim 1 wherein said physical design is an abutted-pin hierarchical physical design including a top-level physical design and a block-level physical design.

11 A method as recited in claim 1 wherein said physical design information includes locations of obstructions of said prior integrated circuit.

12 A method as recited in claim 11 wherein said obstructions include a random access memory (RAM).

13 A method as recited in claim 1 wherein said step c) includes:

partitioning said netlist into a plurality of blocks, each block including a block-level netlist;

performing a top-level floor planning;

performing a top-level placement and route for a plurality of top-level objects;

performing a top-level placement and route for a plurality of ports from said blocks to determine pin assignments for each block; and

generating and optimizing a block-level physical design for each block in parallel

14 A method as recited in claim 13 wherein said generating and optimizing includes:

pressing each portion of each top-level object, which is located within a boundary of a particular block, into said particular block;

generating each pin for each block based on said top-level placement and route to determine pin assignments;

performing a block-level floor planning for each block;

performing a block-level placement for each block;

performing a plurality of block-level operations to optimize each block; and

performing a block-level route for each block

15 A computer-readable medium comprising computer-executable instructions stored therein for performing a method of improving a physical design of a current integrated circuit, said method comprising:

a) receiving a netlist of said current integrated circuit;

b) receiving physical design information from a prior integrated circuit; and;

c) generating said physical design based on said netlist and said physical design information.

16 A computer-readable medium as recited in claim 15 wherein said physical design information includes pin assignments of blocks of said prior integrated circuit.

17 A computer-readable medium as recited in claim 15 wherein said physical design information includes optimal clock distribution tree of said prior integrated circuit.

18 A computer-readable medium as recited in claim 15 wherein said physical design information includes parasitic extraction data of said prior integrated circuit

19 A computer-readable medium as recited in claim 15 wherein said physical design information includes identification of congested blocks of said prior integrated circuit

20 A computer-readable medium as recited in claim 15 wherein said physical design information includes metal resources of said prior integrated circuit.

21 A computer-readable medium as recited in claim 15 wherein said physical design information includes information which facilitates optimizing said current integrated circuit

22 A computer-readable medium as recited in claim 15 wherein said step c) includes:

generating a top-level physical design of said current integrated circuit using said netlist and said physical design information including block-level physical design information of said prior integrated circuit.

23 A computer-readable medium as recited in claim 15 wherein said step c) includes:

generating a block-level physical design of said current integrated circuit using said netlist and said physical design information including block-level physical design information of said prior integrated circuit

24 A computer-readable medium as recited in claim 15 wherein said physical design is an abutted-pin hierarchical physical design including a top-level physical design and a block-level physical design.

25 A computer-readable medium as recited in claim 15 wherein said physical design information includes locations of obstructions of said prior integrated circuit.

26 A computer-readable medium as recited in claim 25 wherein said obstructions include a random access memory (RAM).

27 A computer-readable medium as recited in claim 15 wherein said step c) includes:

partitioning said netlist into a plurality of blocks, each block including a block-level netlist;

performing a top-level floor planning;

performing a top-level placement and route for a plurality of top-level objects;

performing a top-level placement and route for a plurality of ports from said blocks to determine pin assignments for each block; and

generating and optimizing a block-level physical design for each block in parallel

28 A computer-readable medium as recited in claim 27 wherein said generating and optimizing includes:

pressing each portion of each top-level object, which is located within a boundary of a particular block, into said particular block;

generating each pin for each block based on said top-level placement and route to determine pin assignments;

performing a block-level floor planning for each block;

performing a block-level placement for each block;

performing a plurality of block-level operations to optimize each block; and

performing a block-level route for each block

29 A method of determining a plurality of pins for each block of a physical design of a current integrated circuit, comprising:

a) receiving a netlist of said current integrated circuit;

b) receiving physical design information from a prior integrated circuit, wherein said physical design information includes pin assignments of blocks of said prior integrated circuit;

c) using said netlist and said physical design information to perform a top-level placement for a plurality of ports corresponding to each block of said current integrated circuit;

d) using said netlist and said physical design information to perform a top-level route for said ports to determine pin assignments for each block of said current integrated circuit; and

e) generating each pin for each block based on said top-level route to determine pin assignments

30. A method as recited in claim 29 further comprising:

partitioning said netlist into a plurality of blocks of said current integrated circuit, each block including a block-level netlist.

31. A method as recited in claim 29 wherein said physical design information includes optimal clock distribution tree of said prior integrated circuit.

32. A method as recited in claim 29 wherein said physical design information includes parasitic extraction data of said prior integrated circuit.

33. A method as recited in claim 29 wherein said physical design information includes identification of congested blocks of said prior integrated circuit.

34. A method as recited in claim 29 wherein said physical design information includes metal resources of said prior integrated circuit.

35. A method as recited in claim 29 wherein said physical design information includes information which facilitates optimizing said current integrated circuit.

36. A method as recited in claim 29 wherein said physical design information includes locations of obstructions of said prior integrated circuit.

37. A method as recited in claim 36 wherein said obstructions include a random access memory (RAM).

38. A method as recited in claim 29 wherein said physical design is an abutted-pin hierarchical physical design.

39. A method as recited in claim 38 wherein said physical design includes a top-level physical design.

40. A method as recited in claim 38 wherein said physical design includes a block-level physical design.

41. A computer-readable medium comprising computer-executable instructions stored therein for performing a method of determining a plurality of pins for each block of a physical design of a current integrated circuit, comprising:

a) receiving a netlist of said current integrated circuit;

b) receiving physical design information from a prior integrated circuit, wherein said physical design information includes pin assignments of blocks of said prior integrated circuit;

c) using said netlist and said physical design information to perform a top-level placement for a plurality of ports corresponding to each block of said current integrated circuit;

d) using said netlist and said physical design information to perform a top-level route for said ports to determine pin assignments for each block of said current integrated circuit; and

e) generating each pin for each block based on said top-level route to determine pin assignments.

42. A computer-readable medium as recited in claim 41 wherein said method further comprises:

partitioning said netlist into a plurality of blocks of said current integrated circuit, each block including a block-level netlist.

43. A computer-readable medium as recited in claim 41 wherein said physical design information includes optimal clock distribution tree of said prior integrated circuit.

44. A computer-readable medium as recited in claim 41 wherein said physical design information includes parasitic extraction data of said prior integrated circuit.

45. A computer-readable medium as recited in claim 41 wherein said physical design information includes identification of congested blocks of said prior integrated circuit.

46. A computer-readable medium as recited in claim 41 wherein said physical design information includes metal resources of said prior integrated circuit.

47. A computer-readable medium as recited in claim 41 wherein said physical design information includes information which facilitates optimizing said current integrated circuit.

48. A computer-readable medium as recited in claim 41 wherein said physical design information includes locations of obstructions of said prior integrated circuit.

49. A computer-readable medium as recited in claim 48 wherein said obstructions include a random access memory (RAM).

50. A computer-readable medium as recited in claim 41 wherein said physical design is an abutted-pin hierarchical physical design.

51. A computer-readable medium as recited in claim 50 wherein said physical design includes a top-level physical design.

52. A computer-readable medium as recited in claim 50 wherein said physical design includes a block-level physical design.

* * * * *

TAB 20

IW 1491806

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 20, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: ***09/714,722***
FILING DATE: ***November 15, 2000***
PATENT NUMBER: ***6,857,116***
ISSUE DATE: ***February 15, 2005***

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. WALLACE
Certifying Officer

8/Respons
P.Wells
2-21-03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Dahl et al.

Serial No. : 09/714,722 Group Art Unit: 2811

Filed : 11/15/2000 Examiner: TRAN, T.

For : OPTIMIZATION OF ABUTTED-PIN HIERARCHICAL

AMENDMENT AND RESPONSE

Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

In response to the Office Action mailed 10/01/2002, please consider and enter the following arguments for the above captioned patent application. Since February 1, 2003 and February 2, 2003 fall on a weekend, the one month extension period expires on February 3, 2003.

REMARKS

Claims 1-52 were previously pending in this patent application. Claims 1-52 stand rejected. Accordingly, after this Amendment and Response, Claims 1-52 remain pending in this patent application. Further examination and reconsideration in view of the arguments set forth below is respectfully requested.

35 U.S.C. Section 112, First paragraph Rejections

Claims 1-52 stand rejected under 35 U.S.C. Section 112, first paragraph, as based on a disclosure which is not enabling. It was stated that "abutted-pin hierarchical physical design" is critical or essential to the practice of the present invention, but not included in the claims, making the claims not enabled by the disclosure. These rejections are respectfully traversed.

Although it is stated in the Office Action that page 17 of the application teaches that the abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design and provides additional advantages and benefits, it is respectfully asserted that the application does not teach that the "abutted-pin hierarchical physical design" is a "critical" feature as found in In re Mayhew, 527 F. 2d 1229, 188 USPQ 356, (C.C.P.A. 1976). In determining whether a feature is critical, the entire disclosure must be considered. Moreover, features which are merely preferred are not to be considered critical, as noted in In re Goffe, 542 F. 2d 564, 567, 191 USPQ 429, 431 (C.C.P.A. 1976). Limiting an application to the preferred embodiments in the absence of limiting prior art would not serve the

constitutional purpose of promoting the progress in the useful arts. Therefore, an enablement rejection based on the grounds that a disclosed critical limitation is missing from a claim should be made only when the language of the specification makes it clear that the limitation is critical for the invention to function as intended.

In particular, dependent claims 12, 26, 30, and 42 include the limitation "abutted-pin hierarchical physical design". This shows that applicants did not consider the limitation "abutted-pin hierarchical physical design" to be critical, otherwise the Independent Claims 1, 15, 29, and 41 would have included the limitation "abutted-pin hierarchical physical design".

Moreover, at page 13, line 2-4, of the specification, it is clearly stated that the preferred embodiments of the present invention are described in the specification. Additionally, use of the terms "physical design phase" and "physical design information" at page 22 and Figures 9A and 9B demonstrate that the limitation "abutted-pin hierarchical physical design" is not critical to the present invention. In sum, Claims 1-52 are patentable under 35 U.S.C. Section 112, first paragraph, as based on a disclosure which is enabling.

CONCLUSION

It is respectfully submitted that the above remarks and arguments overcome all rejections. For at least the above-presented reasons, it is respectfully submitted that all remaining claims (Claims 1-52) are now in condition for allowance.

The Examiner is urged to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present Application.

Please charge any additional fees or apply any credits to our PTO deposit account number: 23-0085.

Respectfully submitted,

WAGNER, MURABITO & HAO, LLP

Dated: February 3, 2003

Jose S. Garcia

Jose S. Garcia
Registration No. 43,628

Two North Market Street, Third Floor
San Jose, CA 95113
(408) 938-9060

TAB 21

IW 1491806

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 20, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *09/714,722*
FILING DATE: *November 15, 2000*
PATENT NUMBER: *6,857,116*
ISSUE DATE: *February 15, 2005*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

T. WALLACE
Certifying Officer

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/714,722	11/15/2000	Peter Dahl	RESH-001	1564

7590 05/01/2003

WAGNER, MURABITO & HAO LLP
Third Floor
Two North Market Street
San Jose, CA 95113

EXAMINER
TRAN, THIEN F

ART UNIT	PAPER NUMBER
2811	

DATE MAILED: 05/01/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 07-01)

Office Action Summary	Application No. 09/714,722	Applicant(s) DAHL ET AL.
	Examiner Thien Tran	Art Unit 2811

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____ .

2a)☐ This action is FINAL. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-52 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) 1-48 and 50-52 is/are allowed.

6)☒ Claim(s) 49 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

U.S. Patent and Trademark Office PTO-326 (Rev. 04-01) Office Action Summary Part of Paper No. 9

DETAILED ACTION

Drawings

Figure 1 should be designated by a legend such as --Prior Art-- because only that which is old is illustrated. See MPEP § 608.02(g). A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Claim Objections

Claims 6 and 37 are objected to because of the following informalities: lines 1-2, "said obstructions includes" should be --said obstructions include--. Appropriate correction is required.

Claims 20 and 49 are objected to because of the following informalities: line 2, "said obstructions includes" should be --said obstructions include--. Appropriate correction is required.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claim 49 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 49 recites the limitation "said obstructions" in line 2. There is insufficient antecedent basis for this limitation in the claim.

Allowable Subject Matter

Claims 1-48 and 50-52 are allowed.

The following is a statement of reasons for the indication of allowable subject matter: Prior art reference does not teach or renders obvious a method of improving a physical design of a current integrated circuit comprising the steps of receiving physical design information from a prior integrated circuit; and generating said physical design based on a netlist of the current integrated circuit and said physical design information.

Prior art reference does not teach or renders obvious a computer-readable medium comprising computer-executable instructions stored therein for performing a method of improving a physical design of a current integrated circuit, said method comprising receiving physical design information from a prior integrated circuit; and generating said physical design based on a netlist of the current integrated circuit and said physical design information.

Prior art reference does not teach or renders obvious a method of determining a plurality of pins for each block of a physical design of a current integrated circuit comprising receiving physical design information from a prior integrated circuit, wherein said physical design information includes pin assignments of blocks of said prior integrated circuit; using a netlist of said current integrated circuit and said physical design information to perform a top level placement for a plurality of ports corresponding to each block of said current integrated circuit; using said netlist and said physical design information to perform a top level route for said ports to determine pin assignments for each block of said current integrated circuit; and generating each pin for each block based on said top level route to determine pin assignments.

Prior art reference does not teach or renders obvious a computer-readable medium comprising computer-executable instructions stored therein for performing a method of determining a plurality of pins for each block of a physical design of a current integrated circuit comprising receiving physical design information from a prior integrated circuit, wherein said physical design information includes pin assignments of blocks of said prior integrated circuit; using a netlist of said current integrated circuit and said physical design information to perform a top level placement for a plurality of ports corresponding to each block of said current integrated circuit; using said netlist and said physical design information to perform a top level route for said ports to determine pin assignments for each block of said current integrated circuit; and generating each pin for each block based on said top level route to determine pin assignments.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Reference A is being cited since it shows a physical design of an integrated circuit

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thien Tran whose telephone number is (703) 308-4108. The examiner can normally be reached on 8:30AM - 5:00PM Monday through Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tom Thomas can be reached on (703) 308-2772. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 872-9318 for regular communications and (703) 872-9319 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 308-0956.

tt
April 28, 2003

Thien Tran
Patent Examiner
Technology Center 2800

Notice of References Cited	Application/Control No. 09/714,722	Applicant(s)/Patent Under Reexamination DAHL ET AL.	
	Examiner Thien Tran	Art Unit 2811	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-6,425,113	07-2002	Anderson et al.	716/5
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

TAB 22

US006854093B1

(12) **United States Patent**
Dahl et al.

(10) **Patent No.: US 6,854,093 B1**
(45) **Date of Patent: Feb. 8, 2005**

(54) **FACILITATING PRESS OPERATION IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN**

(75) Inventors: **Peter Dahl**, Cupertino, CA (US); **Byron Dickinson**, San Jose, CA (US); **Margie Levine**, Menlo Park, CA (US); **Paul Rodman**, Palo Alto, CA (US)

(73) Assignee: **Reshape, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U S C 154(b) by 279 days

(21) Appl. No : **10/104,813**

(22) Filed: **Mar. 22, 2002**

Related U.S. Application Data

(63) Continuation of application No 09/714,722, filed on Nov 15, 2000.

(51) **Int. Cl.**[7] **G06F 17/50**
(52) **U.S. Cl.** 716/2; 716/3; 716/7; 716/8
(58) **Field of Search** 716/2, 3, 7, 8

(56) **References Cited**

U S PATENT DOCUMENTS

5,475,607	A *	12/1995	Apte et al	716/10
5,519,628	A *	5/1996	Russell et al.	716/10
5,604,680	A *	2/1997	Banji et al	716/8
5,812,415	A *	9/1998	Baisuck	716/11
5,831,869	A *	11/1998	Ellis et al.	716/6
5,943,235	A *	8/1999	Earl et al	700/98
5,970,238	A *	10/1999	Shibata et al.	716/8
6,009,250	A *	12/1999	Ho et al.	716/5
6,047,116	A *	4/2000	Murakami et al	716/19
6,145,117	A *	11/2000	Eng	716/18
6,167,555	A *	12/2000	Lakos	716/3
6,167,556	A *	12/2000	Sun et al.	716/3
6,185,727	B1 *	2/2001	Liebmann	716/19
6,243,854	B1 *	6/2001	Lavin et al	716/19
6,256,768	B1 *	7/2001	Igusa	716/11
6,275,971	B1 *	8/2001	Levy et al	716/5
6,289,493	B1 *	9/2001	Kita	716/11
6,295,633	B1 *	9/2001	Murakawa	716/8
6,301,698	B1 *	10/2001	Lin et al.	716/19
6,327,695	B1 *	12/2001	Bothra et al	716/8
6,343,370	B1 *	1/2002	Taoka et al.	716/21
6,453,452	B1 *	9/2002	Chang et al	716/8
6,507,944	B1 *	1/2003	Kikuchi et al	716/21
6,567,967	B2 *	5/2003	Greidinger et al	716/10
6,647,543	B2 *	11/2003	Yamada et al.	716/21
6,665,857	B2 *	12/2003	Ayres	716/19
2002/0000995	A1 *	1/2002	Sawada et al.	345/620
2002/0188914	A1 *	12/2002	Parashkevov et al	716/3
2003/0018948	A1 *	1/2003	Chang et al	716/8
2003/0046655	A1 *	3/2003	Kimura	716/19

FOREIGN PATENT DOCUMENTS

JP	07140966 A *	6/1995		G09G/5/36

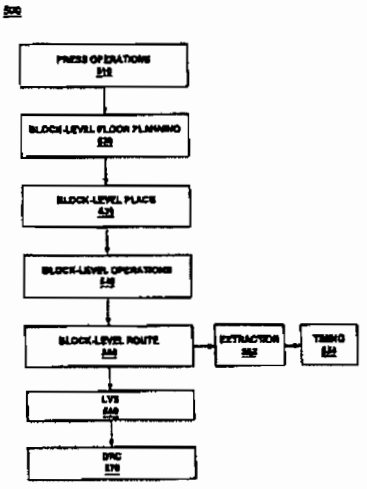
* cited by examiner

Primary Examiner—Matthew Smith
Assistant Examiner—Phallaka Kik
(74) *Attorney, Agent, or Firm*—Wagner, Murabito & Hao LLP

(57) **ABSTRACT**

An abutted-pin hierarchical physical design process is described The abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design and provides additional advantages and benefits. In particular, the abutted-pin hierarchical physical design does not have channels Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced significantly

40 Claims, 31 Drawing Sheets



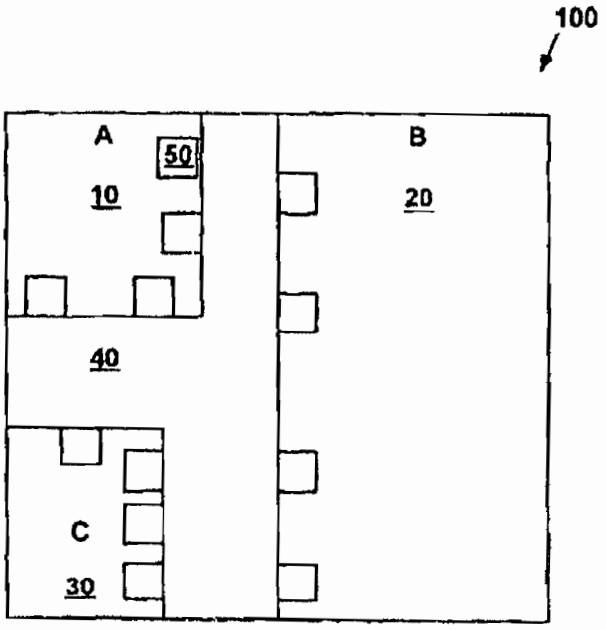


FIGURE 1
(PRIOR ART)

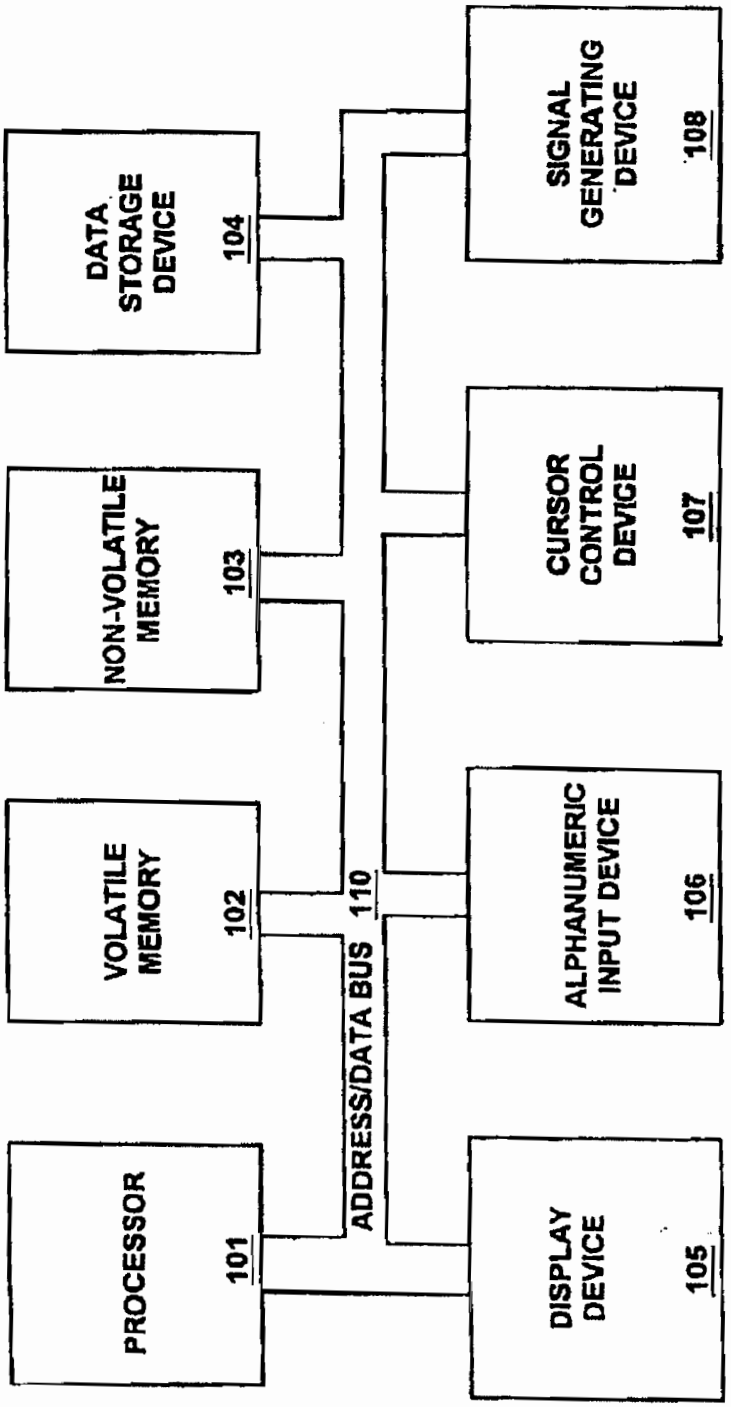

PROCESSOR 101
VOLATILE MEMORY 102
NON-VOLATILE MEMORY 103
DATA STORAGE DEVICE 104
DISPLAY DEVICE 105
ALPHANUMERIC INPUT DEVICE 106
CURSOR CONTROL DEVICE 107
SIGNAL GENERATING DEVICE 108
ADDRESS/DATA BUS 110

FIGURE 2

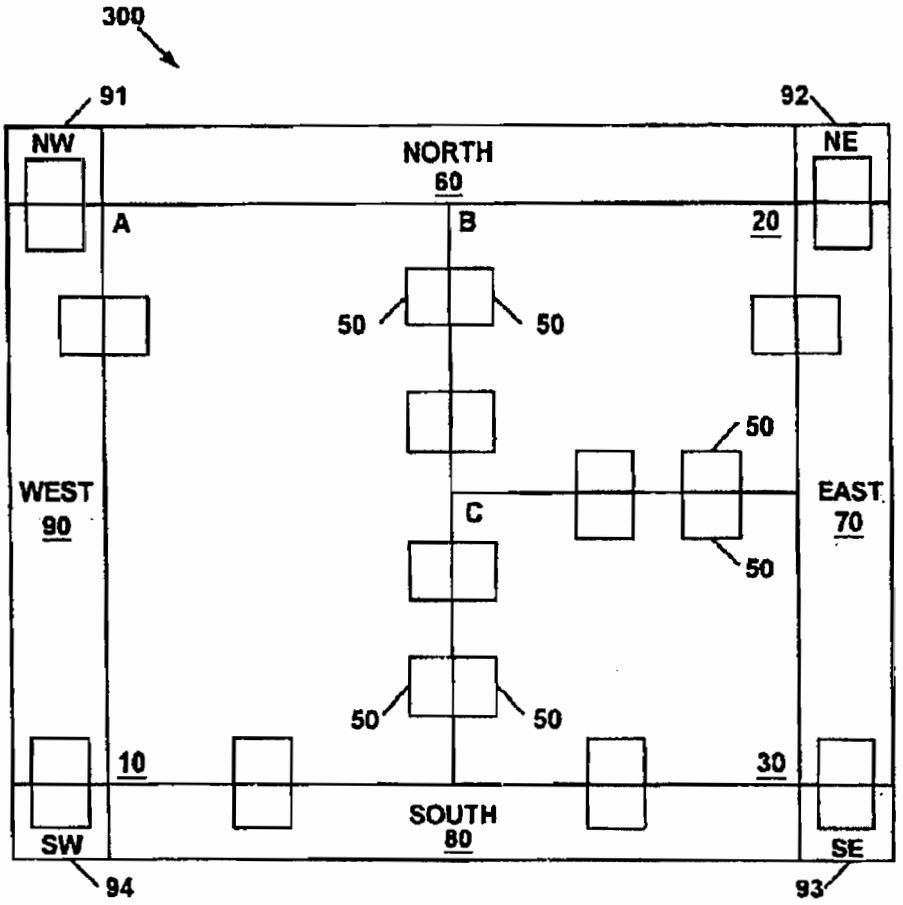

300
91
92
NW
NORTH
60
NE
A
B
20
50
50
50
50
WEST
90
C
EAST
70
50
50
50
10
30
SOUTH
80
SW
SE
94
93

FIGURE 3

400

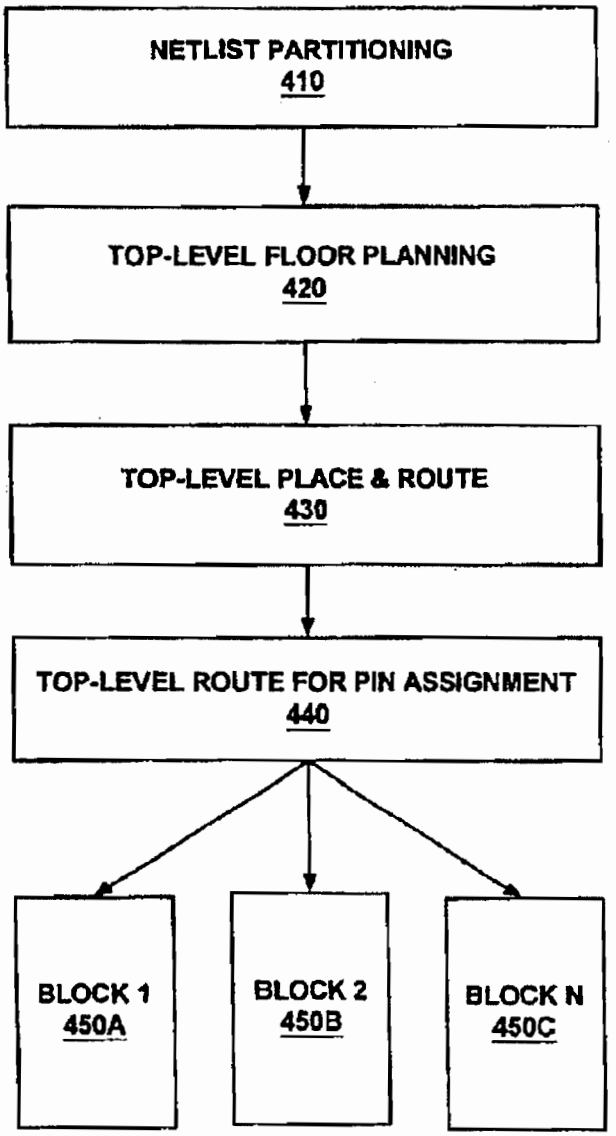

NETLIST PARTITIONING
410
TOP-LEVEL FLOOR PLANNING
420
TOP-LEVEL PLACE & ROUTE
430
TOP-LEVEL ROUTE FOR PIN ASSIGNMENT
440
BLOCK 1
450A
BLOCK 2
450B
BLOCK N
450C

FIGURE 4

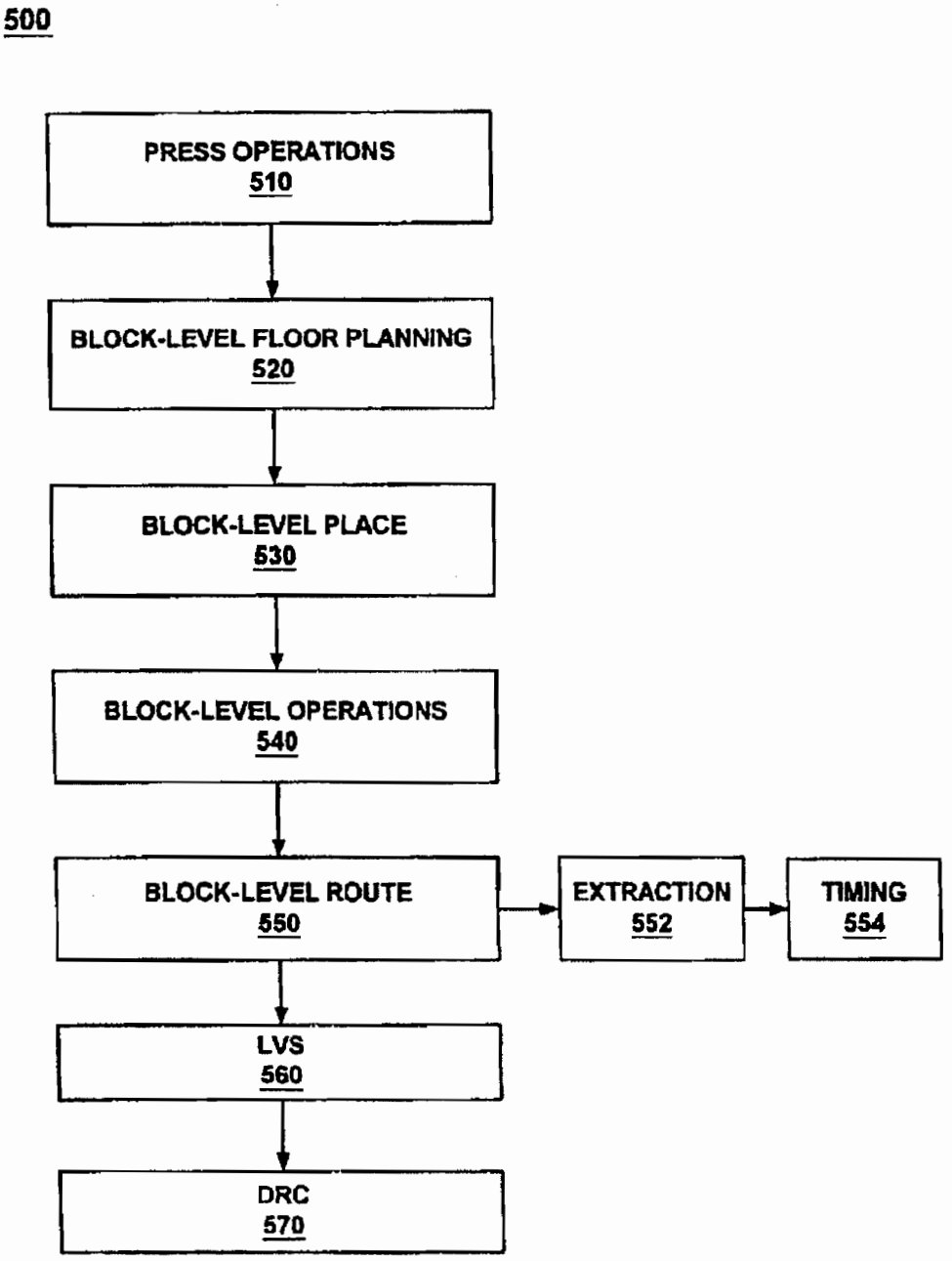

500
PRESS OPERATIONS
510
BLOCK-LEVEL FLOOR PLANNING
520
BLOCK-LEVEL PLACE
530
BLOCK-LEVEL OPERATIONS
540
BLOCK-LEVEL ROUTE
550
EXTRACTION
552
TIMING
554
LVS
560
DRC
570

FIGURE 5

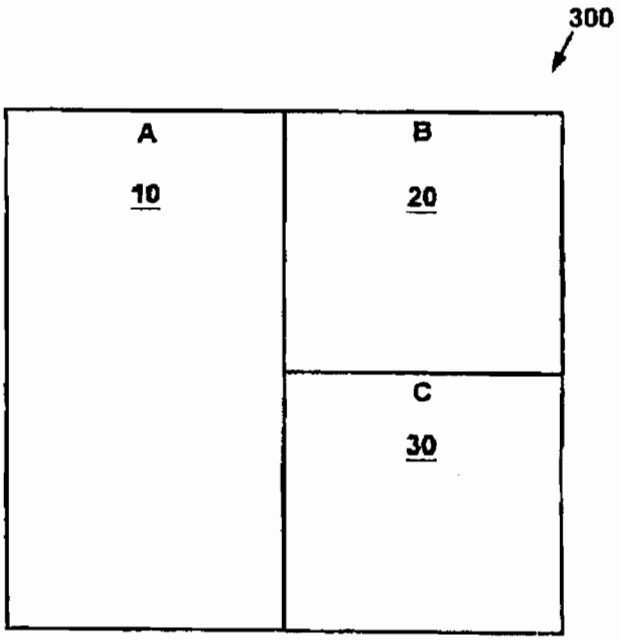


FIGURE 6

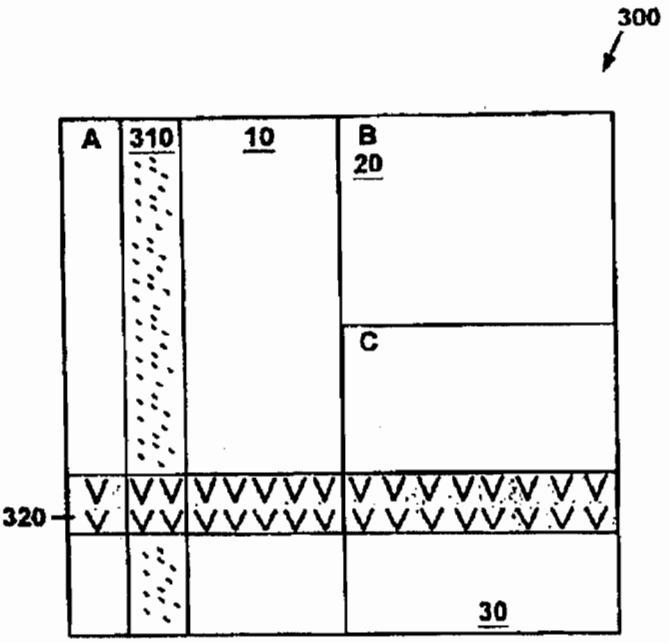


FIGURE 7

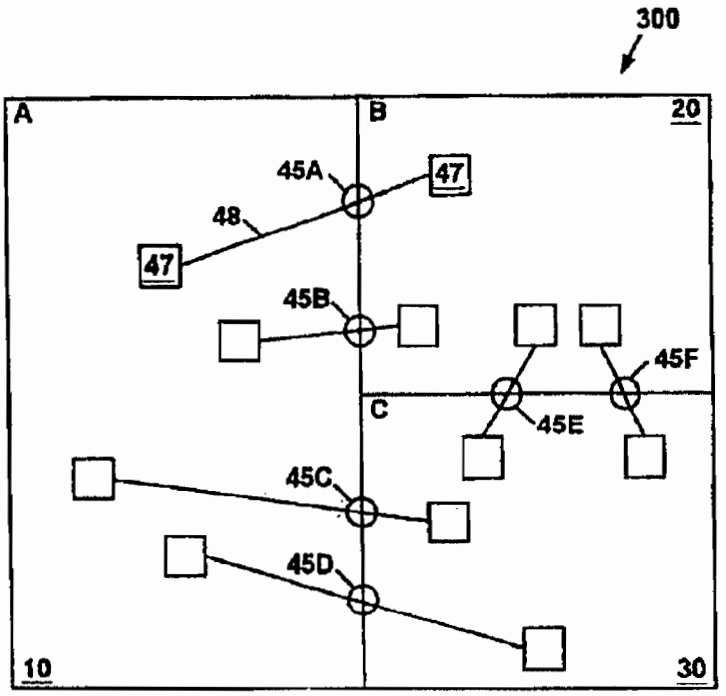


FIGURE 8

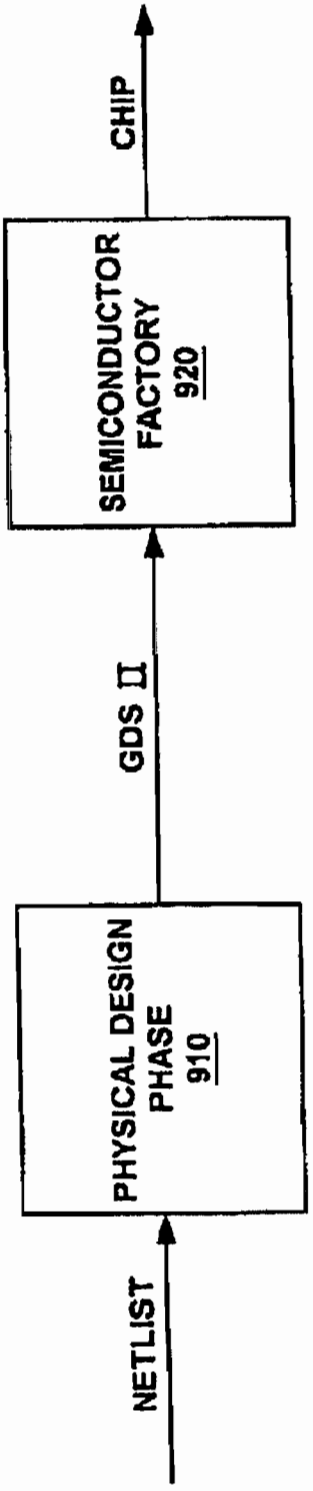


FIGURE 9A
(PRIOR ART)

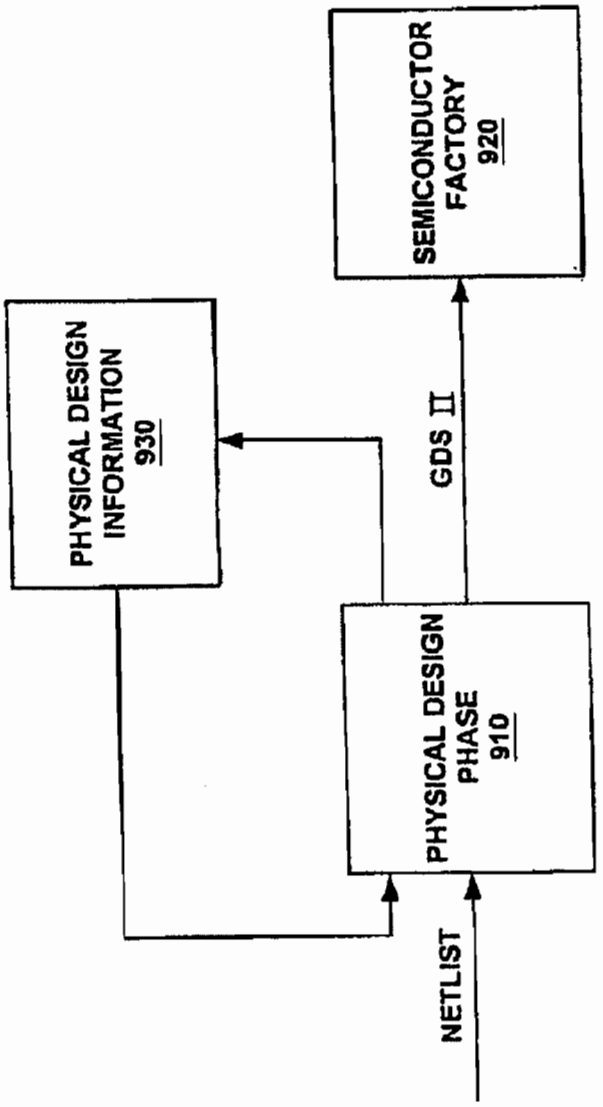

SEMICONDUCTOR FACTORY 920
PHYSICAL DESIGN INFORMATION 930
GDS II
PHYSICAL DESIGN PHASE 910
NETLIST

FIGURE 9B

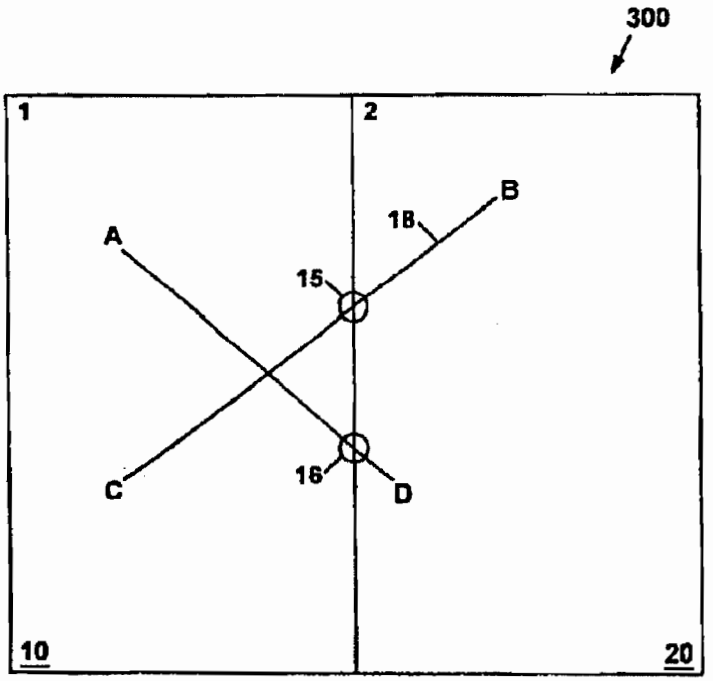


FIGURE 10A

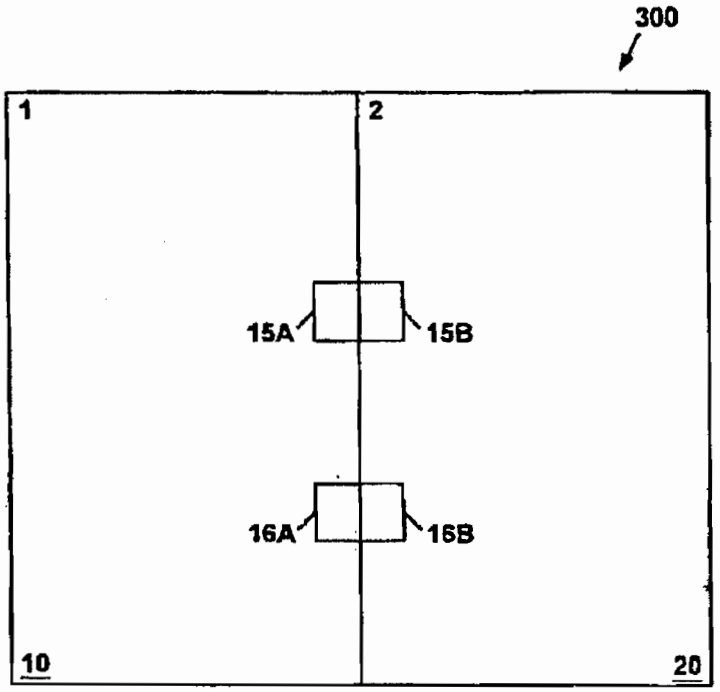


FIGURE 10B

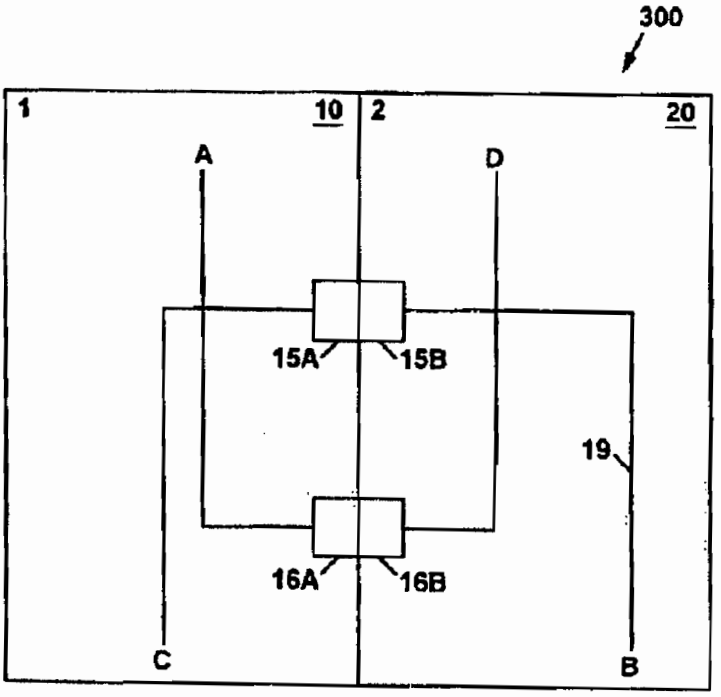


FIGURE 10C

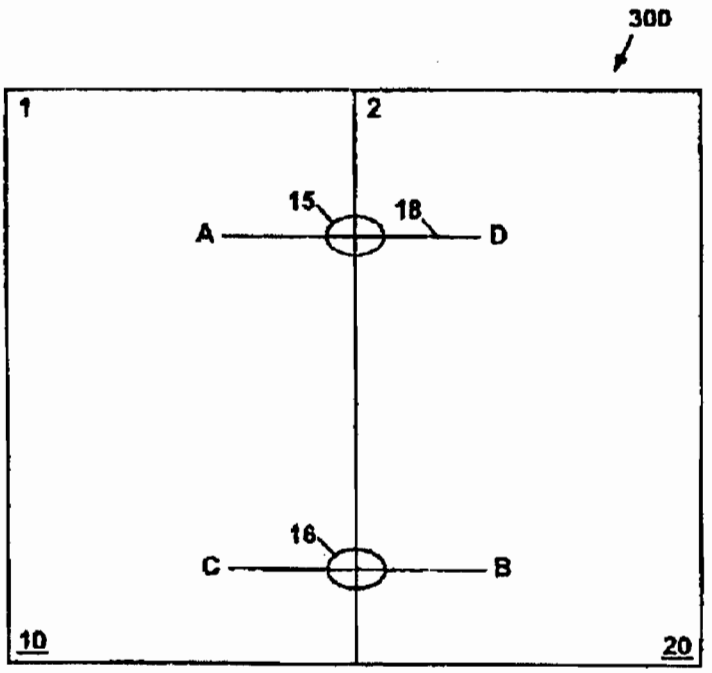


FIGURE 11A

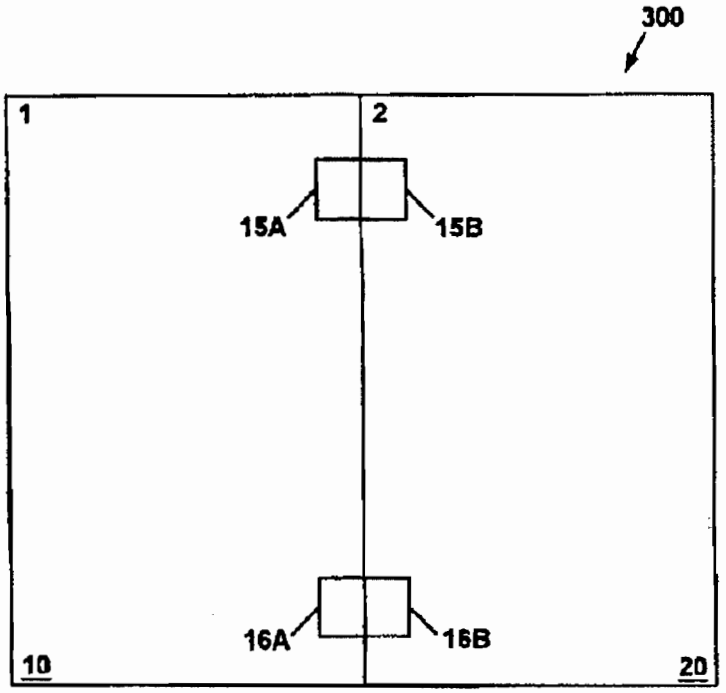


FIGURE 11B

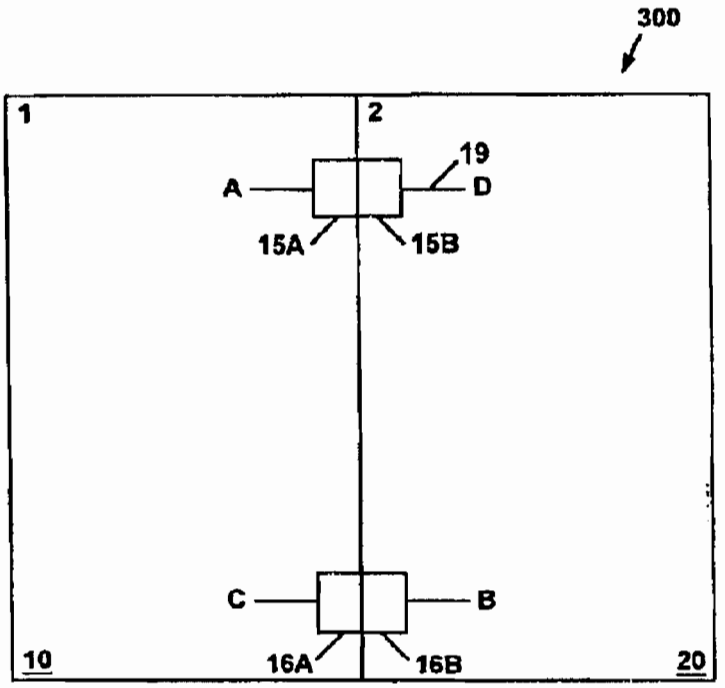


FIGURE 11C

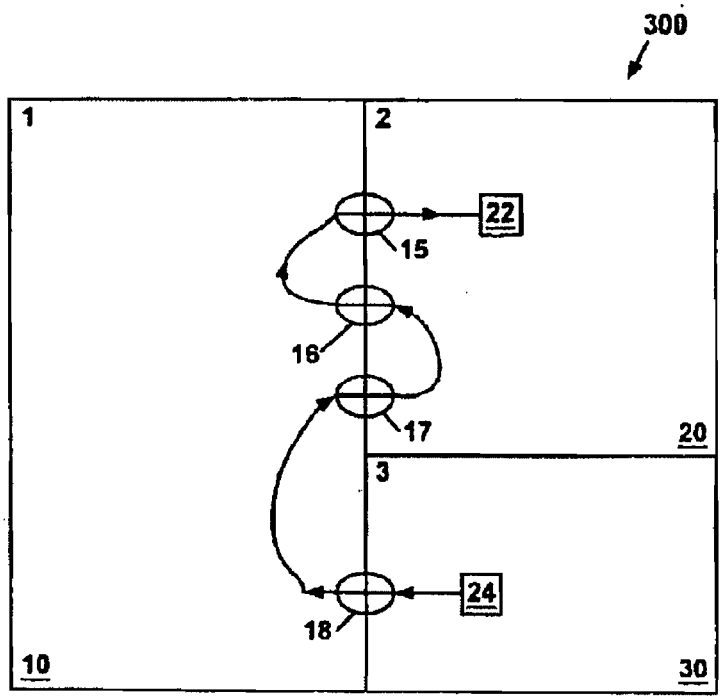


FIGURE 12A

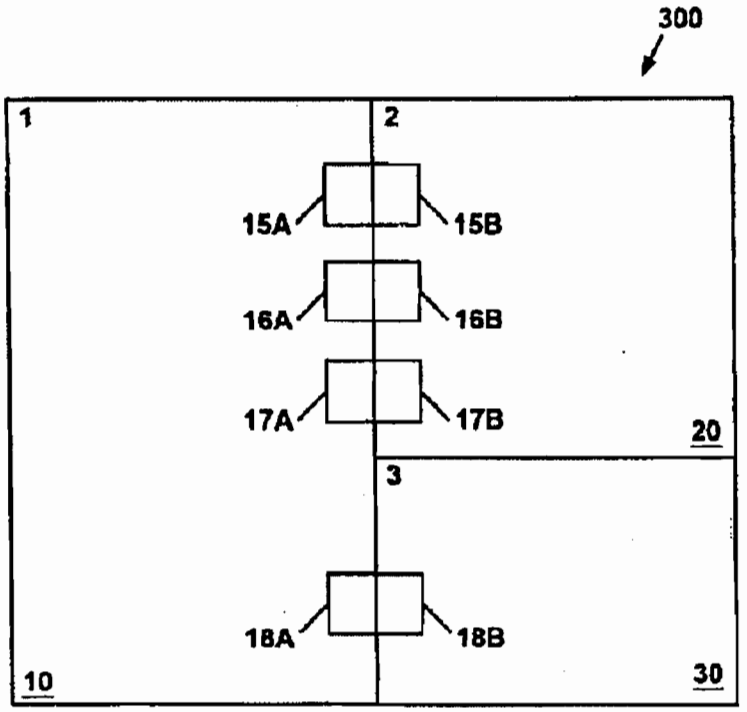


FIGURE 12B

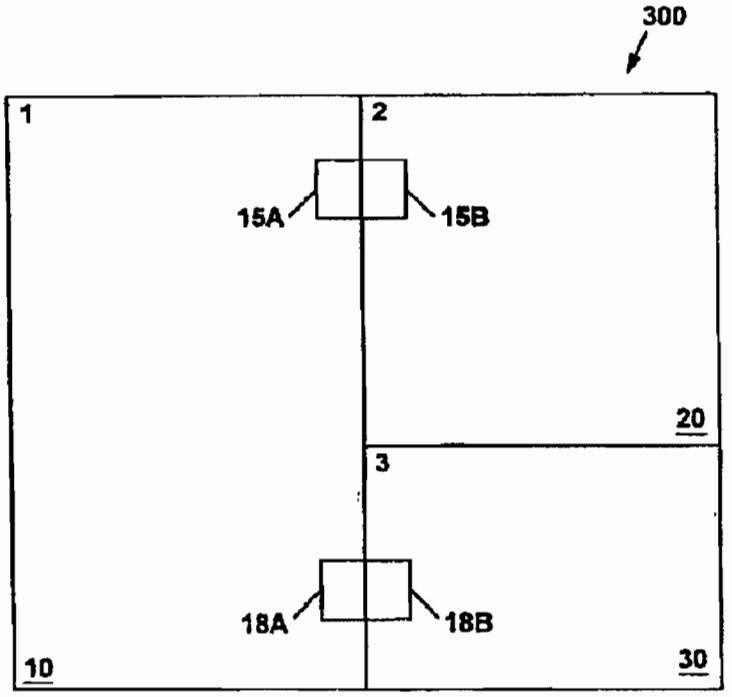


FIGURE 12C

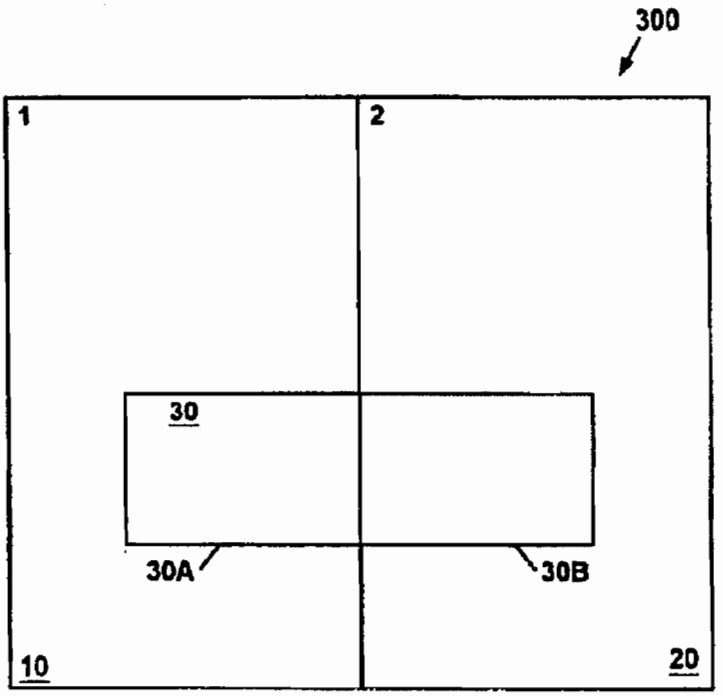


FIGURE 13A

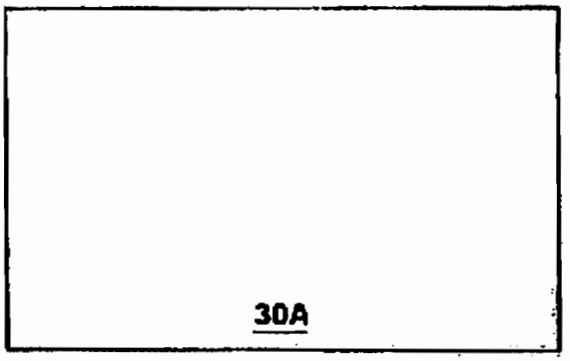


FIGURE 13B

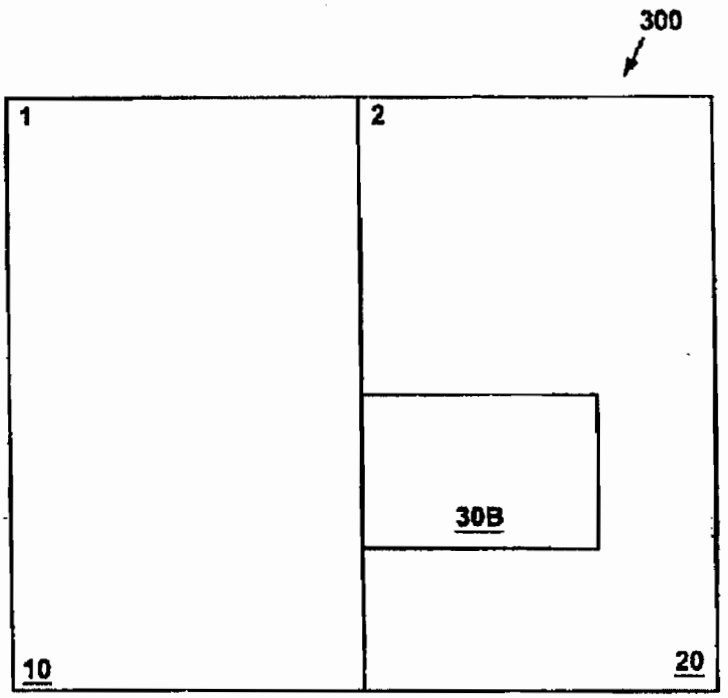


FIGURE 13C

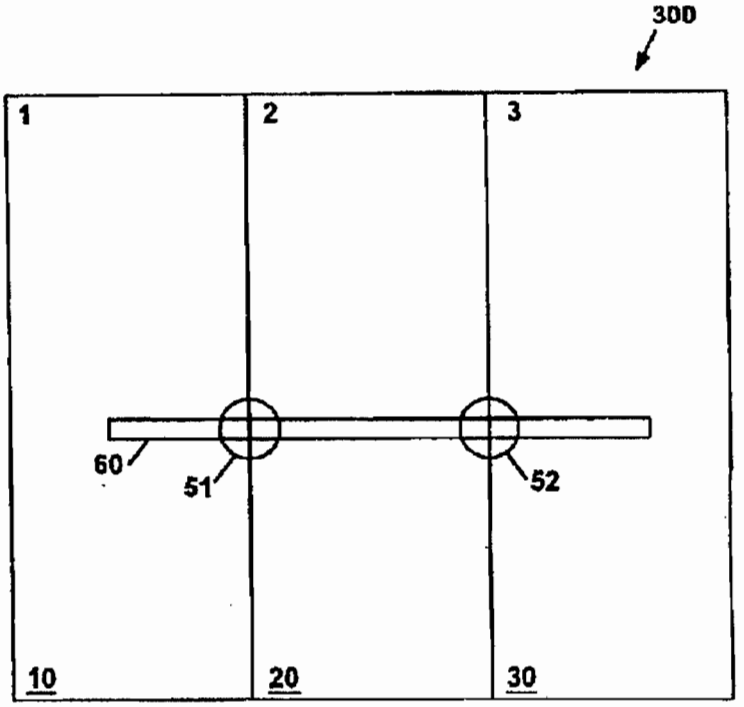


FIGURE 14A

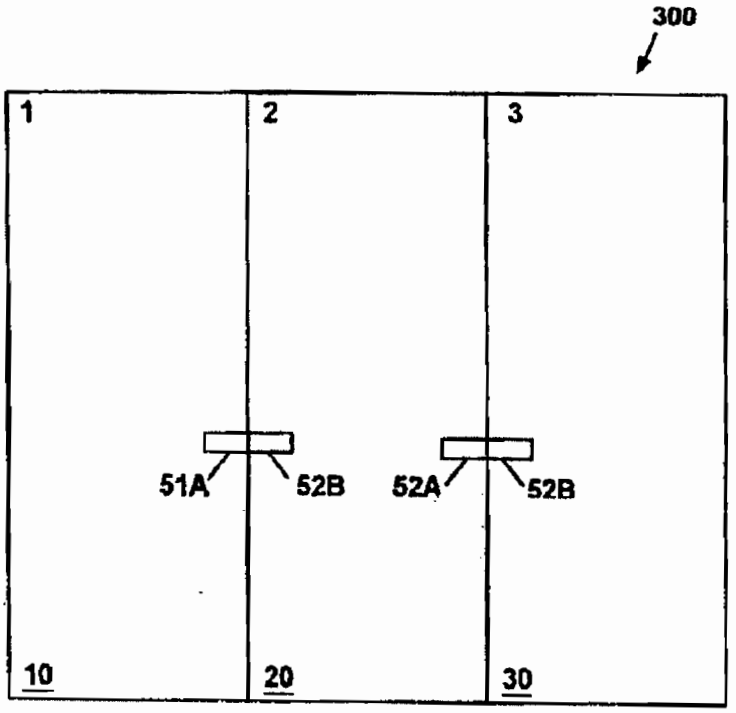


FIGURE 14B

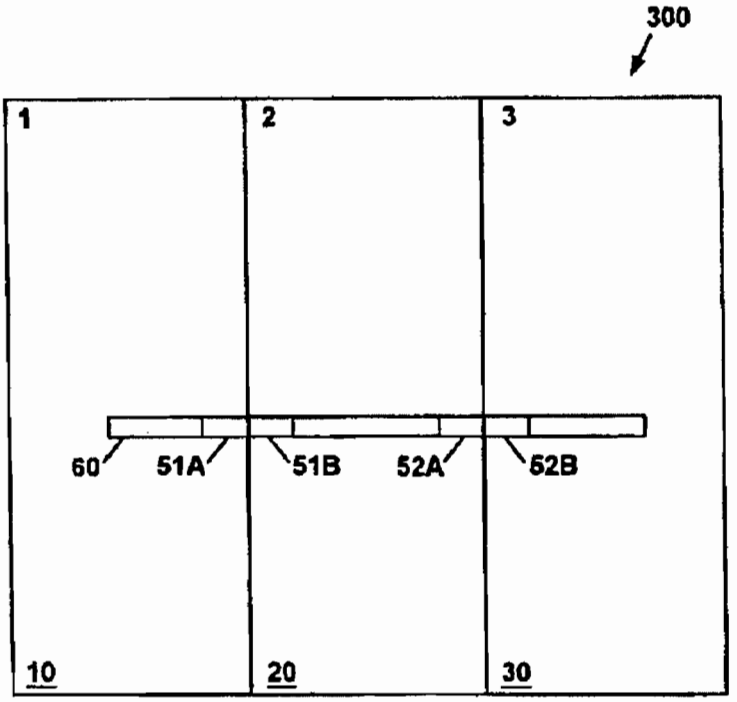


FIGURE 14C

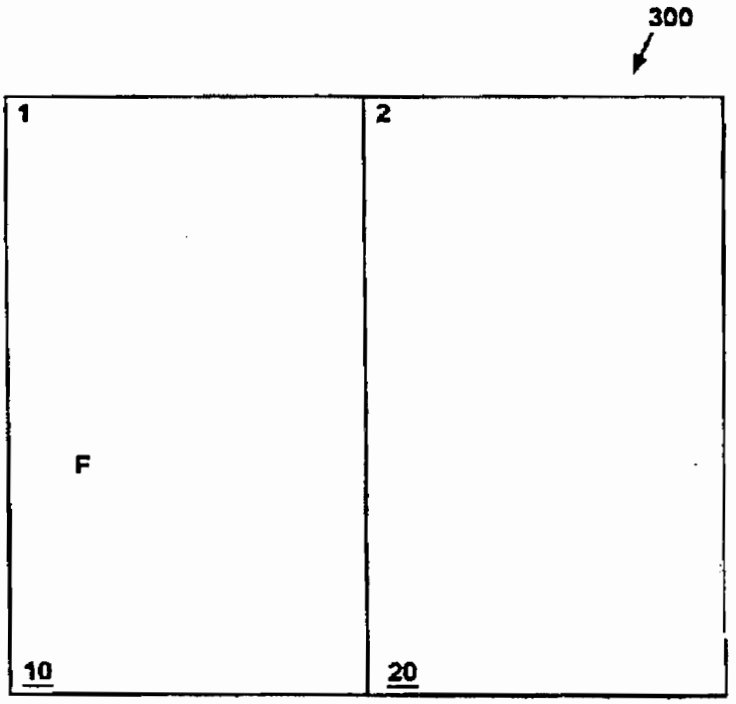


FIGURE 15A

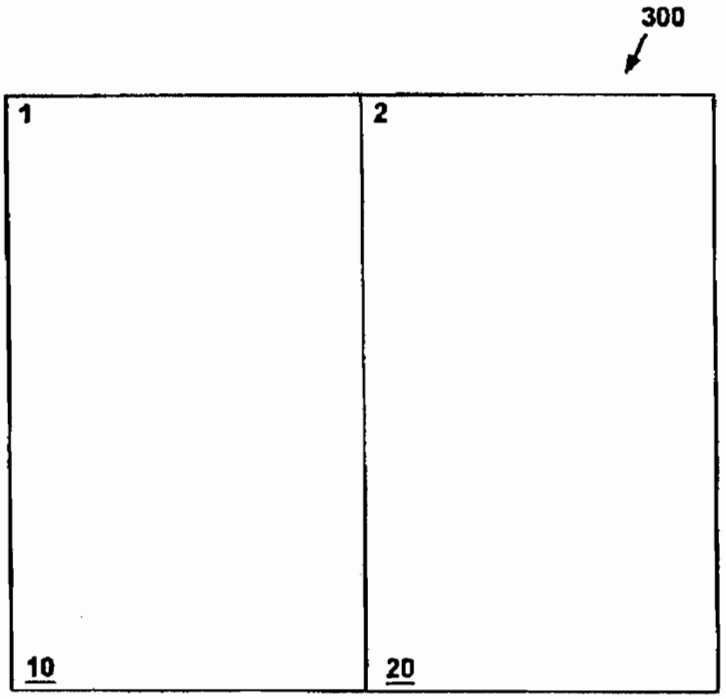


FIGURE 15B

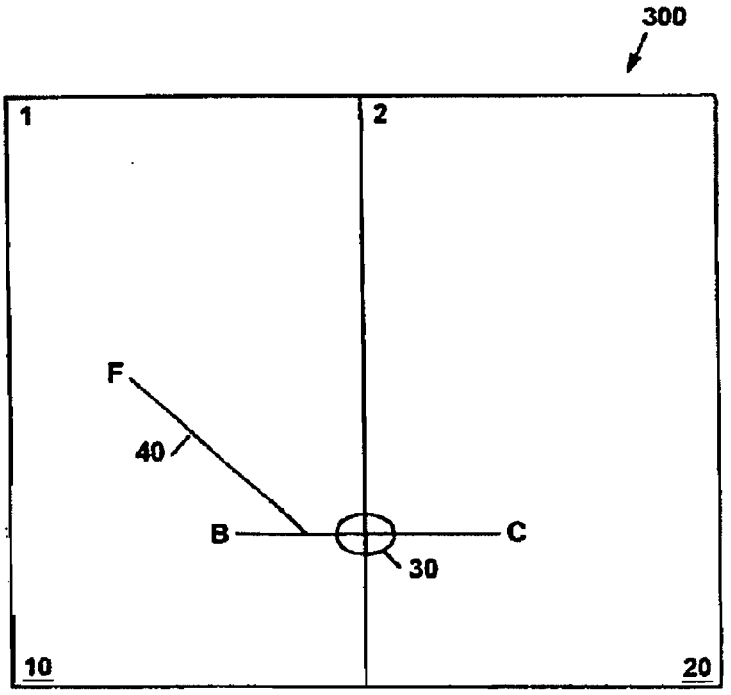


FIGURE 16A

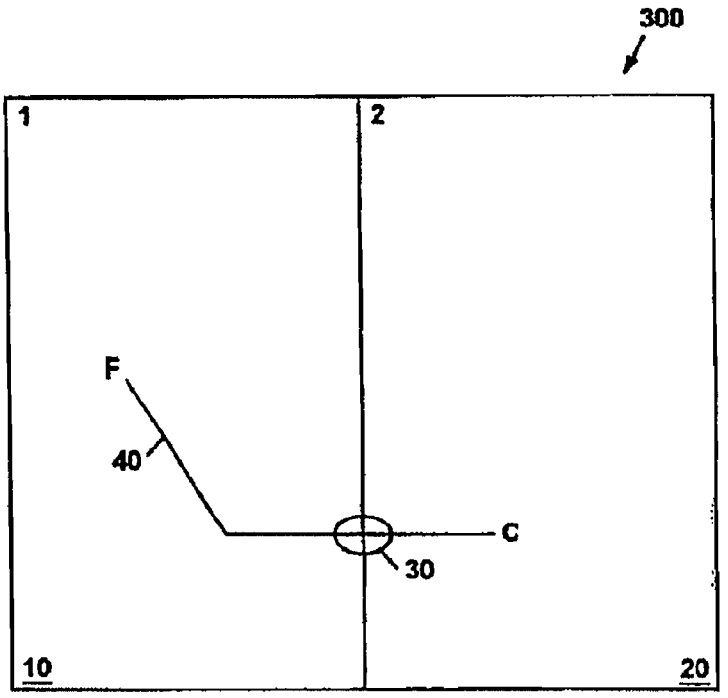


FIGURE 16B

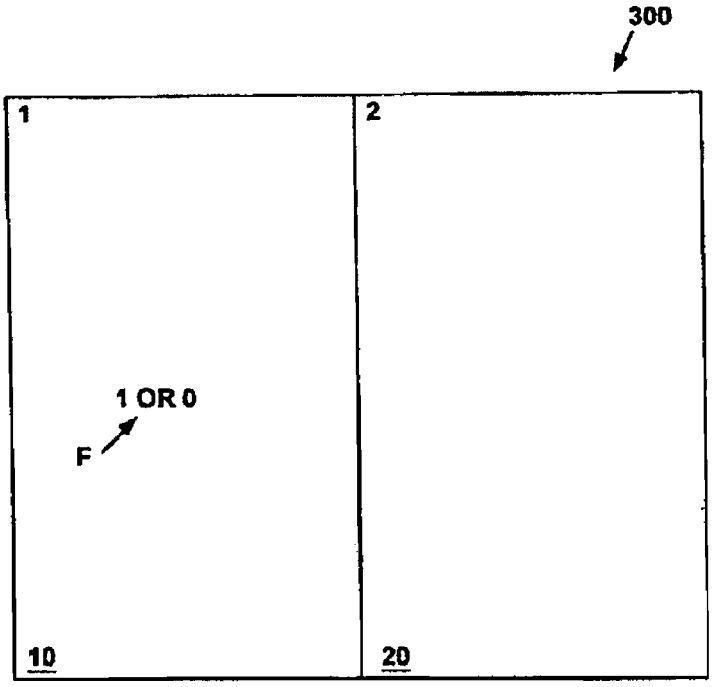


FIGURE 17A

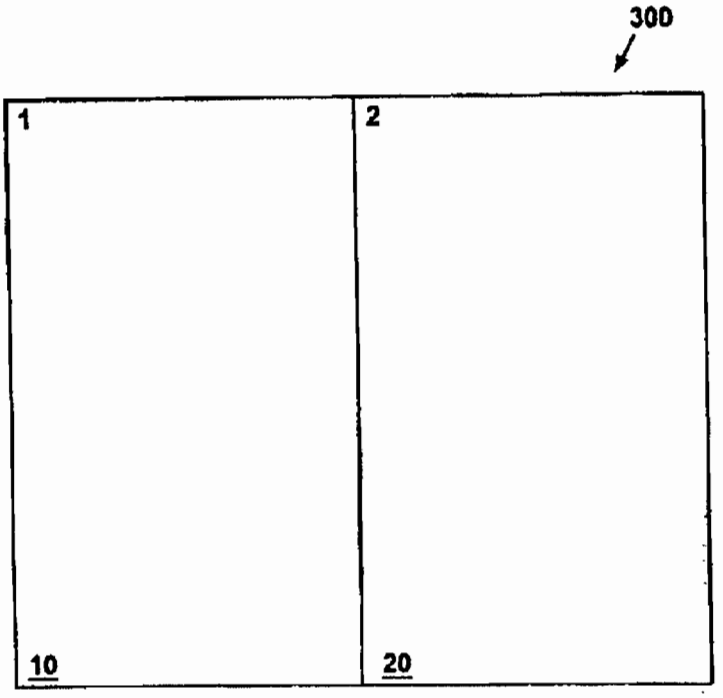


FIGURE 17B

FACILITATING PRESS OPERATION IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN

This patent application is a continuation of application Ser. No 09/714,722, filed Nov 15, 2000, entitled "OPTIMIZATION OF ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN", by Dahl et al, which is hereby incorporated by reference

BACKGROUND OF THE INVENTION

1 Field of the Invention

The present invention generally relates to the field of integrated circuit design More particularly, the present invention relates to the field of software tools for hierarchical physical design

2 Related Art

The tremendous advances in technology have been fueled by improvements in integrated circuit design In particular, integrated circuits have become smaller and more complex. Integrated circuit design engineers depend on electronic design automation (EDA) software tools to facilitate the design of integrated circuits

Typically, the integrated circuit design process begins with a specification which describes the functionality of the integrated circuit and may include a variety of constraints Then, during a logic design phase, the logical implementation of the integrated circuit is determined Several operations are performed to obtain a logical representation of the integrated circuit Generally, EDA software tools use register transfer logic (RTL) to represent the integrated circuit However, additional EDA software tools may be used

After completing the logic design phase, the integrated circuit undergoes a physical design phase Typically, the output of the logic design phase is a netlist, which is then used in the physical design phase Here, EDA software tools layout the integrated circuit to obtain a representation of the physical components in the integrated circuit, whereas the representation indicates the manner in which the integrated circuit will be implemented on a semiconductor chip A variety of operations are performed on the layout of the integrated circuit

At the end of the physical design phase, the representation of the semiconductor chip (in which the integrated circuit is implemented) is sent to a semiconductor manufacturing plant.

Typically, in the physical design phase, EDA software tools implement a flat physical design For example, the components (standard cells, macrocells, etc) of the integrated circuit are placed during a placement operation and are routed during a routing operation However, as the integrated circuit becomes more complex, the EDA software tools struggle to perform the placement operation and the routing operation In particular, the performance of the EDA software tools degrades since the EDA software tools have to manipulate very large files during the placement operation and the routing operation Moreover, as the complexity of the integrated circuit increases, the time necessary to complete the physical design phase increases significantly

Traditional hierarchical physical design has emerged as an alternative to the flat physical design. FIG. 1 illustrates the traditional hierarchical physical design **100**. Here, the components of the integrated circuit are partitioned into a plurality of blocks **10–30** Each block **10–30** includes a plurality of pins **50**, whereas each pin **50** represents a location where a signal can enter the block **10–30** or a location where a signal can exit the block **10–30**. As illustrated in FIG **1**, the traditional hierarchical physical design **100** includes a channel **40**. The channel **40** provides space in order to connect the pins **50** of the blocks **10–30** to one another via metal (not shown) or any other wiring material. The traditional hierarchical physical design **100** enables the placement operation and the routing operation (as well as other operations) for the blocks **10–30** to be performed in parallel with EDA software tools, reducing the time period of the physical design phase. Moreover, the performance of the EDA software tools is improved because the file for each block **10–30** is much smaller than the file for the entire integrated circuit of the flat physical design More importantly, the EDA software tools are better suited to optimize each block **10–30** than to optimize the entire integrated circuit of the flat physical design However, the traditional hierarchical physical design **100** generates wasted space in the channel **40** and generates wiring problems in the channel **40**, such as congestion and crosstalk. Moreover, the traditional hierarchical physical design **100** places and routes components at a top-level (shown in FIG. 1) and a block-level (within each block **10–30**), causing inefficiencies and causing problems with EDA software tools which are configured to operate with flat physical designs

SUMMARY OF THE INVENTION

An abutted-pin hierarchical physical design process is described The abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design and provides additional advantages and benefits. In particular, the abutted-pin hierarchical physical design does not have channels Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced significantly

In the integrated circuit design flow according to an embodiment of the present invention, the physical design phase receives the netlist from the logic design phase In addition, the physical design phase receives physical design information, whereas the physical design information can be any information about a prior integrated circuit that has undergone the physical design phase In an embodiment, the physical design information is stored in a database

In an embodiment of the present invention, the integrated circuit design flow of the present invention is utilized to optimize pin assignment. In an embodiment of the present invention, excess pins formed along a boundary between two blocks are removed

In an embodiment of the present invention, a software tool that performs a "press" operation preserves the properties associated with a segment of a top-level shape despite the shape operation (e.g., AND) being performed with the block and the top-level shape to obtain the segment

If the top-level object has the press property, the top-level object retains its location when the top-level object is "pressed" into a block If the top-level object does not have the press property, the top-level object generally does not retain its location when the top-level object is "pressed" into the block

If in the top-level netlist, the instantiation of a block includes a port that is unused, (thus, not needed for the top-level routing for pin assignment), a software tool removes the port from the top-level netlist, but the block-level netlist of the block remains unchanged

Some software tools are not able to represent the relationship that more than one port is coupled to a pin. Hence, a software tool removes one of the ports from the netlist based on some criteria, such as whether a port is an input port or an output port.

If in the top-level netlist, the instantiation of the block includes a port that is tied to either the power line (1) or the ground line (0) rather to a port of another block, a software tool removes the port from the top-level netlist to avoid routing the port at the top-level. Moreover, the software tool ties the port to either the power line (1) or the ground line (0) in the block-level netlist of the block.

In an embodiment, a software tool performs an unwinding operation which adds to the block-level netlist—of bonding pad blocks—the ports (which were removed earlier by the software tool) that couple to the top-level inputs and to the top-level outputs. Thus, the netlist modified by the physical design phase (e.g., repeater and buffers are added to the netlist) can be compared with the netlist originally received from the logic design phase. In particular, formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification can be performed by software tools.

In an embodiment, each block-level netlist is partitioned into a first netlist and a second netlist. The second netlist and its associated extraction file of each block and the top-level netlist and its associated extraction file are utilized by software tools to perform the timing analysis. This timing analysis can be performed significantly faster than the case where the block-level netlist is not partitioned into the first netlist and the second netlist. In an embodiment, the timing graph resulting from the timing analysis can be analyzed to extract timing constraints (relating to the delay that can be generated by a block) for each block. Hence, if a block is optimized to meet its extracted timing constrains, the block is more likely to meet its timing parameter when the block interacts with the other blocks in the integrated circuit.

These and other advantages of the present invention will no doubt become apparent to those of ordinary skill in the art after having read the following detailed description of the preferred embodiments which are illustrated in the drawing figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the present invention.

FIG. 1 illustrates the traditional hierarchical physical design 100.

FIG. 2 illustrates an exemplary computer system 200 on which embodiments of the present invention may be practiced.

FIG. 3 illustrates an integrated circuit 300 generated with software tools according to an embodiment of the abutted-pin hierarchical physical design process of the present invention.

FIG. 4 illustrates the abutted-pin hierarchical physical design process 400 according to an embodiment of the present invention.

FIG. 5 illustrates the abutted-pin hierarchical physical design process 500 as performed at the block-level in a particular block (450A–450C of FIG. 4) after step 440 of FIG. 4.

FIG. 6 illustrates the layout of the blocks 10–30 is established.

FIG. 7 illustrates a clock wire 320 and a power wire 310 of the top-level.

FIG. 8 illustrates a top-level route for obtaining the pin assignments for each block 10–30.

FIG. 9A illustrates the integrated circuit design flow of the prior art.

FIG. 9B illustrates the integrated circuit design flow according to an embodiment of the present invention.

FIG. 10A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the prior art (FIG. 9A), showing the top-level routing for pin assignment.

FIG. 10B illustrates the integrated circuit 300 of FIG. 10A at the block level.

FIG. 10C illustrates the integrated circuit 300 of FIG. 10B at the block-level.

FIG. 11A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the present invention (FIG. 9B), showing the top-level routing for pin assignment.

FIG. 11B illustrates the integrated circuit 300 of FIG. 11A at the block-level.

FIG. 11C illustrates the integrated circuit 300 of FIG. 11B at the block-level.

FIG. 12A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 12B illustrates the integrated circuit 300 of FIG. 12A at the block-level.

FIG. 12C illustrates the integrated circuit 300 of FIG. 12B, showing the removal of excess pins.

FIG. 13A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 30 (e.g., routing metal).

FIG. 13B illustrates the segment 30A of FIG. 13A.

FIG. 13C illustrates the integrated circuit 300 of FIG. 13A in the top-level, showing that the segment 30A has been removed from the top-level netlist and merged into the block-level netlist of block1 10.

FIG. 14A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 60 (e.g., routing metal).

FIG. 14B illustrates the integrated circuit 300 at the block-level.

FIG. 14C illustrates the integrated circuit 300 at the block-level.

FIG. 15A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 15B illustrates that the port F of block1 10 has been removed from the top-level netlist.

FIG. 16A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment.

FIG. 16B illustrates that the port B of block1 10 has been removed from the netlist for the top-level routing for pin assignment.

FIG. 17A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment

FIG. 17B illustrates that the port F of block1 10 has been removed from the top-level netlist.

The drawings referred to in this description should not be understood as being drawn to scale except if specifically noted

DETAILED DESCRIPTION OF THE INVENTION

Reference will now be made in detail to the preferred embodiments of the present invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with the preferred embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be recognized by one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

Notation and Nomenclature

Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computer memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. In the present application, a procedure, logic block, process, etc., is conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. It has proved convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, a variety of terms are discussed that refer to the actions and processes of an electronic system or a computer system, or other electronic computing device/system. The computer system or similar electronic computing device manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission, or display devices. The present invention is also well suited to the use of other computer systems such as, for example, optical, mechanical, or quantum computers.

Exemplary Computer System Environment

Aspects of the present invention are discussed in terms of steps executed on a computer system. Although a variety of different computer systems can be used with the present invention, an exemplary computer system 200 is shown in FIG. 2.

With reference to FIG. 2, portions of the present invention are comprised of computer-readable and computer executable instructions which reside, for example, in computer-usable media of an electronic system such as the exemplary computer system. FIG. 2 illustrates an exemplary computer system 200 on which embodiments of the present invention may be practiced. It is appreciated that the computer system 200 of FIG. 2 is exemplary only and that the present invention can operate within a number of different computer systems including general-purpose computer systems and embedded computer systems.

Computer system 200 includes an address/data bus 110 for communicating information, a central processor 101 coupled with bus 110 for processing information and instructions, a volatile memory 102 (e.g., random access memory RAM) coupled with the bus 110 for storing information and instructions for the central processor 101 and a non-volatile memory 103 (e.g., read only memory ROM) coupled with the bus 110 for storing static information and instructions for the processor 101. Exemplary computer system 200 also includes a data storage device 104 ("disk subsystem") such as a magnetic or optical disk and disk drive coupled with the bus 110 for storing information and instructions. Data storage device 104 can include one or more removable magnetic or optical storage media (e.g., diskettes, tapes) which are computer readable memories. Memory units of computer system 200 include volatile memory 102, non-volatile memory 103 and data storage device 104.

Exemplary computer system 200 can further include an optional signal generating device 1108 (e.g., a network interface card "NIC") coupled to the bus 1110 for interfacing with other computer systems. Also included in exemplary computer system 200 of FIG. 2 is an optional alphanumeric input device 106 including alphanumeric and function keys coupled to the bus 110 for communicating information and command selections to the central processor 101. Exemplary computer system 200 also includes an optional cursor control or directing device 107 coupled to the bus 110 for communicating user input information and command selections to the central processor 101. An optional display device 105 can also be coupled to the bus 110 for displaying information to the computer user. Display device 105 may be a liquid crystal device, other flat panel display, cathode ray tube, or other display device suitable for creating graphic images and alphanumeric characters recognizable to the user. Cursor control device 107 allows the user to dynamically signal the two-dimensional movement of a visible symbol (cursor) on a display screen of display device 105. Many implementations of cursor control device 107 are known in the art including a trackball, mouse, touch pad, joystick or special keys on alphanumeric input device 106 capable of signaling movement of a given direction or manner of displacement. Alternatively, it will be appreciated that a cursor can be directed and/or activated via input from alphanumeric input device 106 using special keys and key sequence commands.

Abutted-Pin Hierarchical Physical Design

FIG. 3 illustrates an integrated circuit 300 generated with software tools according to the abutted-pin hierarchical physical design process of the present invention. The abutted-pin hierarchical physical design provides solutions to the problems of the traditional hierarchical physical design (see FIG. 1) and provides additional advantages and benefits. In particular, the abutted-pin hierarchical physical design does not have channels. Moreover, in the abutted-pin hierarchical physical design, components of the top-level are merged into the block-level so that the top-level netlist is reduced to instantiations of each block 10–30 and 60–94.

As illustrated in FIG. 3, the abutted-pin hierarchical physical design 300 includes a plurality of blocks 10–30 and 60–94. The netlist of the integrated circuit 300 is partitioned into the plurality of blocks 10–30 and 60–94 such that each block 10–30 and 60–94 has a block level netlist. Blocks 10–30 have the major or core components of the integrated circuit 300. Blocks 60–94 have the bonding pads and other support circuitry of the integrated circuit 300. The blocks 10–30 and 60–94 can be rectangular in shape and can be rectilinear in shape. It should be understood that the integrated circuit 300 can have any number of blocks.

Each block 10–30 and 60–94 has one or more pins 50, whereas each pin 50 represents a location where a signal can enter the block 10–30 and 60–94 or a location where a signal can exit the block 10–30 and 60–94. The edge or boundary of each block 10–30 and 60–94 rests against the edge or boundary of another block 10–30 and 60–94, such that the pin 50 of one block abuts the pin 50 of another block.

Moreover, the top-level components or objects (e.g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc.) are not visible because they have been merged into the blocks 10–30 and 60–94 by a "press" operation performed by a software tool. First, the top-level objects (e.g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc.) are placed and routed at the top-level (the top-level is shown in FIG. 3). In the "press" operation, the top-level objects (e.g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc.) that are within the boundary of a block 10–30 and 60–94 are removed from the top-level netlist and merged into the block-level netlist of that block 10–30 and 60–94. Hence, the abutted-pin hierarchical physical design 300 can be optimized by separately optimizing the individual blocks 10–30 and 60–94. Thus, the software tools can generate (e.g., perform placement, routing, timing, verification, etc.) and optimize the individual blocks 10–30 and 60–94 in parallel. Moreover, a bug within an individual block 10–30 and 60–94 can be corrected by returning that individual block to the logic design phase, while the other blocks continue to undergo the physical design phase.

FIG. 4 illustrates the abutted pin hierarchical physical design process 400 according to an embodiment of the present invention. At 410, a software tool receives the netlist of the integrated circuit from the logic design phase, as described above. The netlist is partitioned into a plurality of blocks, each block having a block-level netlist. In an embodiment, the partitioning of the netlist focuses on reducing the number of ports or terminals of a block that need to couple to the ports or terminals of other blocks.

At 420, a software tool performs top-level floor planning. Here, the layout of each block is determined. At the end of the top-level floor planning, the top-level for an integrated circuit 300 (as shown in FIG. 6) is generated. As illustrated in FIG. 6, the layout of the blocks 10–30 is established. In FIG. 6, the bonding pads 60–94 (of FIG. 3) have been omitted.

At 430, software tools perform top-level placement and routing for the top-level objects (e.g., timing components, clock distribution wiring, power distribution wiring, repeaters, buffers, etc.). FIG. 7 illustrates a clock wire 320 and a power wire 310 of the top-level. The clock wire 320 is routed over BlockA 10 and BlockC 30. The power wire 310 is routed over BlockA 10. It should be understood that any number of additional top-level objects can be placed and routed at the top-level.

At 440, a software tool performs a top-level route for obtaining the pin assignments for each block 10–30, as illustrated in FIG. 8. Since each block 10–30 has one or more ports or terminals 47 that needs to couple to a port or terminal of another block 10–30, the pins for each block 10–30 have to be defined. Initially, the ports 47 of each block 10–30 are placed in a general random location within each block at the top-level since the actual location of the port 47 is not known until a placement operation is performed at the block-level. As illustrated in FIG. 8, the location 45A–45F where a routing wire 48 crosses a boundary between two blocks is defined as a pin for each of the blocks 10–30, facilitating creation of pins that are abutted. In an embodiment, a software tool creates each pin to have a width that is equivalent to the width of the routing wire 48 at the boundary between the two blocks. The pins 50 are illustrated in FIG. 3.

At 450A–450C, the abutted-pin hierarchical physical design process 400 enables software tools to generate and to optimize each block 10–30 in parallel at the block-level.

FIG. 5 illustrates the abutted-pin hierarchical physical design process 500 as performed at the block-level in a particular block (450A–450C of FIG. 4) after step 440 of FIG. 4.

At 510, a software tool performs press operations. The top-level objects illustrated in FIG. 7 (e.g., a clock wire 320 and a power wire 310) and which are located within the boundary of a particular block, are pressed into the particular block. In particular, the top-level objects that are within the boundary of a particular block are removed from the top-level netlist and merged into the block-level netlist of that particular block. Moreover, the pins for the particular block are generated based on the location where the routing wire crosses the boundary between two blocks, as illustrated in FIG. 8 and FIG. 3.

At 520, a software tool performs block-level floor planning for the particular block. At 530, a software tool performs a block-level placement operation for the particular block. At 540, software tools perform a variety of block-level operations to optimize the particular block. Additionally, at 550, a block-level route is performed for the particular block by a software tool. At 552 and 554, software tools perform a block-level extraction operation for determining capacitance and resistance at the nodes and perform block-level timing analysis operations for the particular block.

At 560 and 570, a variety of software tools perform a number of verification operations such as formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification.

FIG. 9A illustrates the integrated circuit design flow of the prior art. As illustrated in FIG. 9A, the physical design phase 910 receives the netlist from the logic design phase (not shown). The physical design phase 910 generates the physi-

cal design for the integrated circuit and outputs a GDS II file. The GDS II file is received by the semiconductor factory 920. The integrated circuit is fabricated by the semiconductor factory 920 on a semiconductor chip.

FIG. 9B illustrates the integrated circuit design flow according to an embodiment of the present invention. As illustrated in FIG. 9B, the physical design phase 910 receives the netlist from the logic design phase (not shown). In addition, the physical design phase 910 receives physical design information 930, whereas the physical design information 930 can be any information about a prior integrated circuit that has undergone the physical design phase 910. In an embodiment, the physical design information 930 is stored in a database. For example, the physical design information 930 can be pin assignments of the prior integrated circuit, optimal clock distribution tree of the prior integrated circuit, parasitic extraction data of the prior integrated circuit, locations of obstructions such as a RAM of the prior integrated circuit, identification of congested blocks of the prior integrated circuit, metal resources for the blocks of the prior integrated circuit, or any other information which can facilitate optimizing the current integrated circuit. Thus, the software tools of the physical design phase 910 can customize the current integrated circuit to avoid the problems of the prior integrated circuit and to realize the benefits of the prior integrated circuit.

In the physical design phase 910, decisions made at the top-level with respect to the top-level objects, significantly influence the creation of problems at the block-level and the optimization operations at the block-level. By using physical design information 930 (concerning the block-level of the prior integrated circuit) at the top-level of the current integrated circuit, the decisions made at the top-level with respect to the top-level objects of the current integrated circuit will be able to reduce the problems present in the prior integrated circuit and will be able to generate solutions to overcome the problems present in the prior integrated circuit, improving the optimization of the abutted-pin hierarchical physical design process of the present invention. Thus, if the physical design information 930 has information about several prior integrated circuits, the current integrated circuit is more likely to be optimized.

In addition, the physical design phase 910 generates the physical design for the integrated circuit and outputs a GDS II file. Moreover, the physical design phase 910 stores physical design information 930 of the current integrated circuit to be used in the physical design phase 910 of a future integrated circuit. The GDS II file is received by the semiconductor factory 920. The integrated circuit is fabricated by the semiconductor factory 920 on a semiconductor chip.

FIG. 10A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the prior art (FIG. 9A), showing the top-level routing for pin assignment. The port C of block1 10 is routed to port B of block2 20. The port A of block1 10 is routed to port D of block 20. This top-level routing has been performed after ports A–D where placed in a generally random location within each block 10–20 at the top-level since the actual locations of the ports A–D are not known until a placement operation is performed at the block-level. Here, the software tools at the top-level do not have access to the physical design information of a prior integrated circuit. The locations 15 and 16 are where the routing metal 18 crosses the boundary between two blocks 10 and 20.

FIG. 10B illustrates the integrated circuit 300 of FIG. 10A. At the block-level, the pins 15A and 16A were formed for block1 10. At the block-level, the pins 15B and 16B were formed for block2 20, whereas pin 15A abuts pin 158 and pin 16A abuts pin 16B. The pins 15A and 15B were formed at location 15 of FIG. 10A. The pins 16A and 16B were formed at location 16 of FIG. 10A.

FIG. 10C illustrates the integrated circuit 300 of FIG. 10B at the block-level. As illustrated in FIG. 10C, the block-level placement operation for block1 10 placed the ports A and C at locations that are different from the locations used to generate the pin assignments in FIG. 10A. In addition, the block-level placement operation for block2 20 placed the ports B and D at locations that are different from the locations used to generate the pin assignments in FIG. 10A. Hence, the block-level routing operations for blocks 10 and 20 generated an inefficient amount of routing wire 19 to couple the ports to the pins in each block. In sum, the pin assignment affects the optimization of the routing wire 19.

FIG. 11A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention and using the integrated circuit design flow of the present invention (FIG. 9B), showing the top-level routing for pin assignment. The port C of block1 10 is routed to port B of block2 20. The port A of block1 10 is routed to port D of block2 20. This top-level routing has been performed after each port A–D where placed in a particular location within each block 10–20 at the top-level, whereas the particular location was based on using the physical design information associated with the prior integrated circuit (FIGS. 1A–10C). Here, the software tools at the top-level have access to the physical design information of the prior integrated circuit (FIGS. 10A–10C). The locations 15 and 16 are where the routing metal 18 crosses the boundary between two blocks 10 and 20.

FIG. 11B illustrates the integrated circuit 300 of FIG. 11A at the block-level. At the block-level, the pins 15A and 16A were formed for block1 10. At the block-level, the pins 15B and 16B were formed for block2 20, whereas pin 15A abuts pin 15B and pin 16A abuts pin 16B. The pins 15A and 15B were formed at location 15 of FIG. 11A. The pins 16A and 16B were formed at location 16 of FIG. 11A. Here, the pins 15A and 15B are associated with ports A and D, unlike FIG. 10B where pins 15A and 15B were associated with ports C and B. Moreover, the pins 16A and 16B of FIG. 11B are associated with ports C and B, unlike FIG. 10B where pins 16A and 16B were associated with ports A and D.

FIG. 11C illustrates the integrated circuit 300 of FIG. 11B at the block-level. As illustrated in FIG. 11C, the block-level placement operation for block1 10 placed the ports A and C at locations that are different from the locations used to generate the pin assignments in FIG. 11A. In addition, the block-level placement operation for block2 20 placed the ports B and D at locations that are different from the locations used to generate the pin assignments in FIG. 11A. However, the difference in the location of the ports between FIG. 11A and FIG. 11C is less than the difference in the location of the ports between FIG. 10A and FIG. 10C. Hence, the block-level routing operations for blocks 10 and 20 generated a more efficient amount of routing wire 19 to couple the ports to the pins in each block, compared to FIG. 10C. In sum, the pin assignments generated with the use of the physical design information of the prior integrated circuit (FIGS. 10A–10C) were more optimal than the pin assignments generated without the use of the physical design information of the prior integrated circuit (FIGS. 10A–10C).

FIG. 12A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the

present invention, showing the top-level routing for pin assignment In the course of routing source port 24 of block3 30 to destination port 22 of block2 20, the software tool that performs the top-level routing for pin assignment crosses the boundary between block1 10 and block2 20 at locations 15,16, and 17, whereas the locations 15,16, and 17 will be defined as pins. The software tool is concerned with routing a path between the source port 24 and the destination port 22, but is not concerned about the number of times the path crosses the boundary between the same blocks

FIG 12B illustrates the integrated circuit 300 of FIG. 12A at the block-level The pins 15A–15B, 16A–16B, and 17A–17B are formed between block1 10 and block2 20 The pins 18A–18B are formed between block1 10 and block3 30. The presence of pins 16A–16B and 17A–17B causes additional routing metal to be added to block1 10 and block2 20 so that pins 15A, 16A, and 17A can be coupled within block1 10 and so that pins 15B, 16B, and 17B can be coupled within block2 20. Hence, one pair of pins (15A–15B or 16A–16B or 17A–17B) is sufficient.

FIG 12C illustrates the integrated circuit 300 of FIG. 12B, showing the removal of excess pins As illustrated in FIG 12C, excess pins 16A–16B and 17A–17B were removed from block1 10 and block2 20. This removal is based on a plurality of criteria, such as the current flow direction between the source port 24 and the destination port 22, the location of the excess pins relative to the source port 24 and the destination port 22, or any other criteria. Here, the criteria kept pins 15A–15B but deleted pins 16A–16B and 17A–17B

FIG. 13A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 30 (e.g., routing metal) As described above, a software tool performs a press operation so that the portion of the top-level object 30 which is within the boundary of a particular block 10–20 is moved from the top-level netlist to the block-level netlist of the particular block 10–20 In particular, the segment 30A is pressed into block1 10 while the segment 30B is pressed into block2 20 In an embodiment, the shape operations of a database are utilized in performing the press operation. In FIG. 13A, an AND operation would be performed with block1 10 and the shape 30 to obtain the segment 30A (FIG. 13B) Typically, the routing metal 30 includes a plurality of properties that are stored in a database These properties identify the routing metal 30 and describe the function of the routing metal 30. However, in the shape operations (e.g., AND) of the prior art, the shape operation returns the segment 30A (FIG 13B) without its properties Thus, these properties have to be reconstructed

In the present invention, the software tool that performs the press operation preserves the properties associated with segment 30A of the routing metal 30 despite the shape operation (e g , AND) performed with block1 10 and the shape 30 to obtain the segment 30A (FIG 13B)

FIG 13C illustrates the integrated circuit 300 of FIG. 13A in the top-level, showing that the segment 30A has been removed from the top-level netlist and merged into the block-level netlist of block1 10 Moreover, the properties associated with segment 30A at the top-level are transferred to the segment 30A at the block-level

FIG 14A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for a top-level object 60 (e g , routing metal) As illustrated in FIG 14A, the top-level object 60 is routed through block1 10, block2 20, and block3 30. The locations 51–52 indicate top-level object 60 crosses a boundary between two blocks In an embodiment, a press property is added to the properties of the top-level object 60 stored in a database. If the top-level object 60 has the press property, the top-level object 60 retains its location when the top-level object 60 is pressed into block1 10, block2 20, and block3 30, as illustrated in the block-level view of the integrated circuit 300 in FIG 14C. If the top-level object 60 does not have the press property, the top-level object 60 generally does not retain its location when the top-level object 60 is pressed into block1 10, block2 20, and block3 30, as illustrated in the block-level view of the integrated circuit 300 in FIG. 14B. For example, top-level objects such as power and ground have the press property. As illustrated in FIG 14B, the pins 51A–51B and 52A–52B are defined. However, the software tool is not constrained to placing the top-level object 60 in the block-level exactly as it was placed at the top-level. Moreover, the top-level object is placed in the block-level of block1 10, block2 20, and block3 30 according to the separate placement and routing requirements of block1 10, block2 20, and block3 30.

FIG. 15A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment As illustrated in FIG 15A, in the top-level netlist, the instantiation of block1 10 includes a port F that is unused, thus, not needed for the top-level routing for pin assignment Hence, a software tool removes port F from the top-level netlist, but the block-level netlist of block1 10 remains unchanged In an embodiment, the software tool that performs the press operation removes the port F FIG. 15B illustrates that the port F of block1 10 has been removed from the top-level netlist

FIG. 16A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment As illustrated in FIG. 16A, port F and port B of block1 10 are coupled to port C of block2 20 with a routing metal 40. However, at location 30 the routing metal 40 crosses the boundary between block1 10 and block2 20. If a pin is formed within block1 10 at location 30, the pin would be coupled to port F and to port B However, some software tools are not able to represent this relationship (i e , more than one port coupled to a pin) Hence, a software tool removes one of the ports (port F or port B) from the netlist based on some criteria, such as whether a port is an input port or an output port FIG 16B illustrates that the port B of block1 10 has been removed from the netlist for the top-level routing for pin assignment

FIG. 17A illustrates an integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention, showing the top-level routing for pin assignment As illustrated in FIG 17A, in the top-level netlist, the instantiation of block1 10 includes a port F that is tied to either the power line (1) or the ground line (0) rather to a port of another block Hence, a software tool removes port F from the top-level netlist to avoid routing the port F at the top-level Moreover, the software tool ties the port F to either power line (1) or the ground line (0) in the block-level netlist of block1 10 FIG 17B illustrates that the port F of block1 10 has been removed from the top-level netlist.

As illustrated in FIG 3, the integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention includes a North bond pad block 60, an

East bond pad block **70**, a South bond pad block **80**, and a West bond pad block **90**, each having bond pad cells The top-level netlist of the integrated circuit **300** includes one or more top-level inputs for receiving external signals and one or more top-level outputs for transmitting signals off the chip The top-level inputs and the top-level outputs are coupled to bond pad cells Typically, software tools which perform a routing operation are configured to not perform the routing operation if the netlist includes bond pad cells Since the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** have bond pad cells in the block-level netlist, the software tools refuse to perform the routing operation in these blocks, preventing pins to be formed on the boundary between these blocks and the blocks **10–30** (the core blocks)

In the present invention, the bond pad cells are marked as macrocells rather than bond pad cells, allowing pins to be formed on the boundary between these blocks **60**, **70**, **80**, and **90** and the blocks **10–30** (the core blocks)

Typically, the block-level netlist of the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** include nets to the top-level inputs and nets to the top-level outputs Generally, the block-level-netlist of the North bond pad block **60**, the East bond pad block **70**, the South bond pad block **80**, and the West bond pad block **90** include nets to the bond pad cells

In an embodiment of the present invention, a software tool removes the nets to the top-level inputs and nets to the top-level outputs so that the physical design of the integrated circuit can be accomplished as described above In an embodiment, the software tool removes in the block-level netlist the ports that couple to the top-level inputs and to the top-level outputs Moreover, the software tool adds a property to the nets to the bond pad cells to indicate that these nets are suppose to couple to the top-level inputs and to the top-level outputs, facilitating an unwinding operation to rep establish at the block-level netlist the nets to the top-level inputs and nets to the top-level outputs that were removed earlier The unwinding operation adds to the block-level netlist the ports (which were removed earlier) that couple to the top-level inputs and to the top-level outputs Thus, the netlist modified by the physical design phase (e g , repeater and buffers are added to the netlist) can be compared with the netlist originally received from the logic design phase In particular, formal verification, layout versus schematic (LVS) verification, and design rules check (DRC) verification can be performed by software tools

A challenge with implementing an integrated circuit based on the abutted-pin hierarchical physical design process of the present invention involves analyzing the timing of signal paths that traverse more than one block The timing of these global paths is difficult to analyzed compared to analyzing the timing of local paths, whereas local paths are signal paths that do not leave a block One method of analyzing the timing of these global paths involves partitioning the block-level netlist of each block into a first netlist and a second netlist The first netlist includes nets which start at a register (or flip-flop) and end at a register (or flip-flop) within the block, whereas each branch of the net also starts at a register (or flip-flop) and ends at a register (or flip-flop) within the block The second netlist includes nets which are coupled to a pin of the block Generally, the first netlist is ¾ of the initial block-level netlist while the second netlist is ¼ of the initial block-level netlist If the second netlist ratio is greater than ¼, this indicates inefficient partitioning of the blocks

Once the first netlist and the second netlist are obtain, an extraction operation to obtain parasitic resistance and capacitance is performed on the second netlist of each block In an embodiment, the partitioning of the block-level netlist and the extraction operation in each block are performed in parallel Moreover, an extraction operation is performed on the top-level netlist In an embodiment, a software tool replaces the abutted pins of the top-level netlist with zero ohm resistors

Some software tools utilized to perform the timing analysis are unable to operate on netlists having nets that are coupled to multiple pins of a block In an embodiment of the present invention, these netlist are transformed by using "assign statements" to assign different names to the nets that are coupled to multiple pins of a block Hence, each different named net can be coupled to a separate pin of the block

In an embodiment, the second netlist and its associated extraction file of each block and the top-level netlist and its associated extraction file are utilized by software tools to perform the timing analysis This timing analysis can be performed significantly faster than the case where the block-level netlist is not partitioned into the first netlist and the second netlist In an embodiment, the timing graph resulting from the timing analysis can be analyzed to extract timing constraints (relating to the delay that can be generated by a block) for each block Hence, if a block is optimized to meet its extracted timing constrains, the block is more likely to meet its timing parameter when the block interacts with the other blocks in the integrated circuit

The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated It is intended that the scope of the invention be defined by the claims appended hereto and their equivalents

What is claimed is:

1 A method of pressing a top-level object into one or more blocks of a physical design, comprising:

a) performing a shape operation to identify a portion of said top-level object that is within a boundary of a particular block, wherein said top-level object includes a plurality of properties, and wherein said shape operation preserves said properties associated with said portion of said top-level object; and

b) moving said portion of said top-level object and said properties associated with said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state

2 A method as recited in claim **1** wherein said physical design is an abutted-pin hierarchical physical design

3 A method as recited in claim **2** wherein said physical design includes a top-level physical design

4 A method as recited in claim **2** wherein said physical design includes a block-level physical design

5 A method as recited in claim **1** wherein said top-level object is a timing component

6 A method as recited in claim **1** wherein said top-level object is a clock distribution wiring

7. A method as recited in claim 1 wherein said top-level object is a power distribution wiring.

8. A method as recited in claim 1 wherein said top-level object is a routing metal

9. A method as recited in claim 1 wherein said shape operation is an AND operation

10. A method as recited in claim 1 wherein said properties are stored in a database.

11. A computer-readable medium comprising computer-executable instructions stored therein for performing a method of pressing a top-level object into one or more blocks of a physical design, said method comprising:

a) performing a shape operation to identify a portion of said top-level object that is within a boundary of a particular block, wherein said top-level object includes a plurality of properties, and wherein said shape operation preserves said properties associated with said portion of said top-level object; and

b) moving said portion of said top-level object and said properties associated with said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state.

12. A computer-readable medium as recited in claim 11 wherein said physical design is an abutted-pin hierarchical physical design

13. A computer-readable medium as recited in claim 12 wherein said physical design includes a top-level physical design

14. A computer-readable medium as recited in claim 12 wherein said physical design includes a block-level physical design

15. A computer-readable medium as recited in claim 11 wherein said top-level object is a timing component.

16. A computer-readable medium as recited in claim 11 wherein said top-level object is a clock distribution wiring.

17. A computer-readable medium as recited in claim 11 wherein said top-level object is a power distribution wiring

18. A computer-readable medium as recited in claim 11 wherein said top-level object is a routing metal.

19. A computer-readable medium as recited in claim 11 wherein said shape operation is an AND operation

20. A computer-readable medium as recited in claim 11 wherein said properties are stored in a database

21. A method of pressing a top-level object into one or more blocks of a physical design, comprising:

a) identifying a portion of said top-level object that is within a boundary of a particular block;

b) moving said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state;

c) if said top-level object includes a press property, performing a block-level placement for said particular block such that a block-level physical location of said portion of said top-level object is substantially equivalent to a top-level physical location of said portion of said top-level object; and

d) if said top-level object does not include a press property, performing said block-level placement for said particular block without regard to said top-level physical location of said portion of said top-level object

22. A method as recited in claim 21 wherein said physical design is an abutted-pin hierarchical physical design

23. A method as recited in claim 22 wherein said physical design includes a top-level physical design.

24. A method as recited in claim 22 wherein said physical design includes a block-level physical design

25. A method as recited in claim 21 wherein said top-level object is a timing component

26. A method as recited in claim 21 wherein said top-level object is a clock distribution wiring

27. A method as recited in claim 21 wherein said top-level object is a power distribution wiring.

28. A method as recited in claim 21 wherein said top-level object is a routing metal.

29. A method as recited in claim 21 wherein said top-level object is a ground distribution wiring

30. A method as recited in claim 21 wherein said press property is stored in a database

31. A computer-readable medium comprising computer-executable instructions stored therein for performing a method of pressing a top-level object into one or more blocks of a physical design, said method comprising:

a) identifying a portion of said top-level object that is within a boundary of a particular block;

b) moving said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state;

c) if said top-level object includes a press property, performing a block-level placement for said particular block such that a block-level physical location of said portion of said top-level object is substantially equivalent to a top-level physical location of said portion of said top-level object; and

d) if said top-level object does not include a press property, performing said block-level placement for said particular block without regard to said top-level physical location of said portion of said top-level object.

32. A computer-readable medium as recited in claim 31 wherein said physical design is an abutted-pin hierarchical physical design

33. A computer-readable medium as recited in claim 32 wherein said physical design includes a top-level physical design

34. A computer-readable medium as recited in claim 32 wherein said physical design includes a block-level physical design.

35. A computer-readable medium as recited in claim 31 wherein said top-level object is a timing component

36. A computer-readable medium as recited in claim 31 wherein said top-level object is a clock distribution wiring

37. A computer-readable medium as recited in claim 31 wherein said top-level object is a power distribution wiring.

38. A computer-readable medium as recited in claim 31 wherein said top-level object is a routing metal.

39. A computer-readable medium as recited in claim 31 wherein said top-level object is a ground distribution wiring.

40. A computer-readable medium as recited in claim 31 wherein said press property is stored in a database.

* * * * *

TAB 23

IW 1491804

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

July 21, 2006

THIS IS TO CERTIFY THAT ANNEXED IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF THE FILE WRAPPER AND CONTENTS OF:

APPLICATION NUMBER: *10/104,813*
FILING DATE: *March 22, 2002*
PATENT NUMBER: *6,854,093*
ISSUE DATE: *February 08, 2005*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

E. BORNETT
Certifying Officer

2825
41

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Dahl et al.

Serial No. : 10/104,813 Group Art Unit: 2825

Filed : 03/22/2002 Examiner: KIK, P.

Title: FACILITATING PRESS OPERATION IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN

AMENDMENT AND RESPONSE

Commissioner for Patents
P.O Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Office Action mailed 04/09/2004, please enter and consider the following amendments and arguments for the above captioned patent application.

Amendments to the Specification begin at page 3 of this paper.

Amendments to the Claims are reflected in the listing of claims which begins on page 9 of this paper.

Amendments to the Drawings begin at page 17 of this paper and include both an attached replacement sheet and an annotated sheet showing changes.

Remarks/Arguments begin on page 18 of this paper.

An **Appendix** including amended drawing figures is attached at the end of this paper.

Amendments to the Specification:

Please replace paragraph beginning at page 15, line 16, with the following amended paragraph:

Computer system 200 includes an address/data bus [[100]] 110 for communicating information, a central processor 101 coupled with bus [[100]] 110 for processing information and instructions, a volatile memory 102 (e.g., random access memory RAM) coupled with the bus 110 for storing information and instructions for the central processor 101 and a non-volatile memory 103 (e.g., read only memory ROM) coupled with the bus 110 for storing static information and instructions for the processor 101. Exemplary computer system 200 also includes a data storage device 104 ("disk subsystem") such as a magnetic or optical disk and disk drive coupled with the bus 110 for storing information and instructions. Data storage device 104 can include one or more removable magnetic or optical storage media (e.g., diskettes, tapes) which are computer readable memories. Memory units of computer system 200 include volatile memory 102, non-volatile memory 103 and data storage device 104.

Please replace paragraph beginning at page 18, line 1, with the following amended paragraph:

Each block 10-30 and 60-94 has one or more pins 50, whereas each pin 50 represents a location where a signal can enter the block 10-30 and 60-94 or a location where a signal can exit the block 10-30 and 60-94. The edge or boundary of each block 10-30 and 60-94 rests against the edge or boundary of another block 10-30 and 60-94, such that the pin 50 of one block abuts the pin 50 of another block.

Please replace paragraph beginning at page 21, line 6, with the following amended paragraph:

At [[510]] 520, a software tool performs block-level floor planning for the particular block. At 530, a software tool performs a block-level placement operation for the particular block. At 540, software tools perform a variety of block-level operations to optimize the particular block. Additionally, at [[540]] 550, a block-level route is performed for the particular block by a software tool. At 552 and 554, software tools perform a block-level extraction operation for determining capacitance and resistance at the nodes and perform block-level timing analysis operations for the particular block.

Please replace paragraph beginning at page 22, line 1, with the following amended paragraph:

Figure 9B illustrates the integrated circuit design flow according to an embodiment of the present invention. As illustrated in Figure [[9A]] 9B, the physical design phase 910 receives the netlist from the logic design phase (not shown). In addition, the physical design phase 910 receives physical design information 930 , whereas the physical design information 930 can be any information about a prior integrated circuit that has undergone the physical design phase 910. In an embodiment, the physical design information 930 is stored in a database. For example, the physical design information 930 can be pin assignments of the prior integrated circuit, optimal clock distribution tree of the prior integrated circuit, parasitic extraction data of the prior integrated circuit, locations of obstructions such as a RAM of the prior integrated circuit, identification of congested blocks of the prior integrated circuit, metal resources for the blocks of the prior integrated circuit, or any other information which can facilitate optimizing the current integrated circuit. Thus, the software tools of the physical design phase 910 can customize the current integrated circuit to avoid the problems of the prior integrated circuit and to realize the benefits of the prior integrated circuit.

Please replace paragraph beginning at page 25, line 9, with the following amended paragraph:

Figure 11B illustrates the integrated circuit 300 of Figure 11A at the block-level. At the block-level, the pins 15A and 16A were formed for block1 10. At the block-level, the pins 15B and 16B were formed for block2 20, whereas pin 15A abuts pin 15B and pin 16A abuts pin 16B. The pins 15A and 15B were formed at location 15 of Figure 11A. The pins 16A and 16B were formed at location 16 of Figure 11A. Here, the pins 15A and [[15A]] 15B are associated with ports A and D, unlike Figure 10B where pins 15A and 15B were associated with ports C and B. Moreover, the pins 16A and [[16A]] 16B of Figure 11B are associated with ports C and B, unlike Figure 10B where pins 16A and 16B were associated with ports A and D.

Please replace paragraph beginning at page 27, line 6, with the following amended paragraph:

Figure 12C illustrates the integrated circuit 300 of Figure 12B, showing the removal of excess pins. As illustrated in Figure 12C, excess pins 16A-16B and 17A-17B were removed from block1 10 and block2 20. This removal is based on a plurality of criteria, such as the current flow direction between the source port 24 and the destination port 22, the location of the excess pins relative to the source port 24 and the destination port 22, or any other criteria. Here, the criteria kept pins 15A-15B but deleted pins 16A-16B and 17A-17B.

Please replace paragraph beginning at page 31, line 1, with the following amended paragraph:

As illustrated in Figure 3, the integrated circuit 300 based on the abutted-pin hierarchical physical design process of the present invention includes a North bond pad block 60, an East bond pad block 70, a South bond pad block 80, and a West bond pad block 90, each having bond pad cells. The top-level netlist of the integrated circuit 300 includes one or more top-level inputs for receiving external signals and one or more top-level outputs for transmitting ~~signal~~ signals off the chip. The top-level inputs and the top-level outputs are coupled to bond pad cells. Typically, software tools which perform a routing operation are configured to not perform the routing operation if the netlist includes bond pad cells. Since the North bond pad block 60, the East bond pad block 70, the South bond pad block 80, and the West bond pad block 90 have bond pad cells in the block-level netlist, the software tools refuse to perform the routing operation in these blocks, preventing pins to be formed on the boundary between these blocks and the blocks 10-30 (the core blocks).

Please replace paragraph beginning at page 31, line 15, with the following amended paragraph:

In the present invention, the bond pad cells are marked as macrocells rather than bond pad cells, allowing pins to be formed on the boundary between these blocks 60, 70, 80, and 90 and the blocks 10-30 (the core blocks).

Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (Cancelled)

2. (Currently Amended) A method of pressing a top-level object into one or more blocks of a physical design, comprising:

a) performing a shape operation to identify a portion of said top-level object that is within a boundary of a particular block, wherein said top-level object includes a plurality of properties, and wherein said shape operation preserves said properties associated with said portion of said top-level object; and

b) moving said portion of said top-level object and said properties associated with said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state.

3. (Original) A method as recited in Claim 2 wherein said physical design is an abutted-pin hierarchical physical design.

4. (Original) A method as recited in Claim 3 wherein said physical design includes a top-level physical design.

5. (Original) A method as recited in Claim 3 wherein said physical design includes a block-level physical design.

6. (Original) A method as recited in Claim 2 wherein said top-level object is a timing component.

7. (Original) A method as recited in Claim 2 wherein said top-level object is a clock distribution wiring.

8. (Original) A method as recited in Claim 2 wherein said top-level object is a power distribution wiring.

9. (Original) A method as recited in Claim 2 wherein said top-level object is a routing metal.

10. (Original) A method as recited in Claim 2 wherein said shape operation is an AND operation.

11. (Original) A method as recited in Claim 2 wherein said properties are stored in a database.

12. (Currently Amended) A computer-readable medium comprising computer-executable instructions stored therein for performing a method of pressing a top-level object into one or more blocks of a physical design, said method comprising:

a) performing a shape operation to identify a portion of said top-level object that is within a boundary of a particular block, wherein said top-level object includes a plurality of properties, and wherein said shape operation preserves said properties associated with said portion of said top-level object; and

b) moving said portion of said top-level object and said properties associated with said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state.

13. (Original) A computer-readable medium as recited in Claim 12 wherein said physical design is an abutted-pin hierarchical physical design.

14. (Original) A computer-readable medium as recited in Claim 13 wherein said physical design includes a top-level physical design.

15. (Original) A computer-readable medium as recited in Claim 13 wherein said physical design includes a block-level physical design.

16. (Original) A computer-readable medium as recited in Claim 12 wherein said top-level object is a timing component.

17. (Original) A computer-readable medium as recited in Claim 12 wherein said top-level object is a clock distribution wiring.

18. (Original) A computer-readable medium as recited in Claim 12 wherein said top-level object is a power distribution wiring.

19. (Original) A computer-readable medium as recited in Claim 12 wherein said top-level object is a routing metal.

20. (Original) A computer-readable medium as recited in Claim 12 wherein said shape operation is an AND operation.

21. (Original) A computer-readable medium as recited in Claim 12 wherein said properties are stored in a database.

22. (Currently Amended) A method of pressing a top-level object into one or more blocks of a physical design, comprising:

a) identifying a portion of said top-level object that is within a boundary of a particular block;

b) moving said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state;

c) if said top-level object includes a press property, performing a block-level placement for said particular block such that a block-level physical location of said portion of said top-level object is substantially equivalent to a top-level physical location of said portion of said top-level object; and

d) if said top-level object does not include a press property, performing said block-level placement for said particular block without regard to said top-level physical location of said portion of said top-level object.

23. (Original) A method as recited in Claim 22 wherein said physical design is an abutted-pin hierarchical physical design.

24. (Original) A method as recited in Claim 23 wherein said physical design includes a top-level physical design.

25. (Original) A method as recited in Claim 23 wherein said physical design includes a block-level physical design.

26. (Original) A method as recited in Claim 22 wherein said top-level object is a timing component.

27. (Original) A method as recited in Claim 22 wherein said top-level object is a clock distribution wiring.

28. (Original) A method as recited in Claim 22 wherein said top-level object is a power distribution wiring.

29. (Original) A method as recited in Claim 22 wherein said top-level object is a routing metal.

30. (Original) A method as recited in Claim 22 wherein said top-level object is a ground distribution wiring.

31. (Original) A method as recited in Claim 22 wherein said press property is stored in a database.

32. (Currently Amended) A computer-readable medium comprising computer-executable instructions stored therein for performing a method of pressing a top-level object into one or more blocks of a physical design, said method comprising:

a) identifying a portion of said top-level object that is within a boundary of a particular block;

b) moving said portion of said top-level object from a top-level netlist to a block-level netlist of said particular block so that said top-level object changes from a flattened state to a hierarchical state;

c) if said top-level object includes a press property, performing a block-level placement for said particular block such that a block-level physical location of said portion of said top-level object is substantially equivalent to a top-level physical location of said portion of said top-level object; and

d) if said top-level object does not include a press property, performing said block-level placement for said particular block without regard to said top-level physical location of said portion of said top-level object.

33. (Original) A computer-readable medium as recited in Claim 32 wherein said physical design is an abutted-pin hierarchical physical design.

34. (Original) A computer-readable medium as recited in Claim 33 wherein said physical design includes a top-level physical design.

35. (Original) A computer-readable medium as recited in Claim 33 wherein said physical design includes a block-level physical design.

36. (Original) A computer-readable medium as recited in Claim 32 wherein said top-level object is a timing component.

37. (Original) A computer-readable medium as recited in Claim 32 wherein said top-level object is a clock distribution wiring.

38. (Original) A computer-readable medium as recited in Claim 32 wherein said top-level object is a power distribution wiring.

39. (Original) A computer-readable medium as recited in Claim 32 wherein said top-level object is a routing metal.

40. (Original) A computer-readable medium as recited in Claim 32 wherein said top-level object is a ground distribution wiring.

41. (Original) A computer-readable medium as recited in Claim 32 wherein said press property is stored in a database.

Amendments to the Drawings:

The attached sheets of drawings include changes to Figure 1 and Figure 11A. One sheet, which includes Figure 1, replaces the sheet including Figure 1. In Figure 1, the label "PRIOR ART" has been added. A second sheet, which includes Figure 11A, replaces the sheet including Figure 11A. In Figure 11A, a previously omitted horizontal line has been added.

Attachment: Replacement Sheet for Figure 1

Annotated Sheet Showing Changes for Figure 1

Replacement Sheet for Figure 11A

Annotated Sheet Showing Changes for Figure 11A

TITLE: OPTIMIZATION OF THE TOP LEVEL IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN
Inventor (s): Peter Dahl, Byron Dickinson, Margie Levine, Paul Rodman
USSN: 10/104,813 Attorney Docket #: RESH-001.CON1

Replacement Sheet 1/31

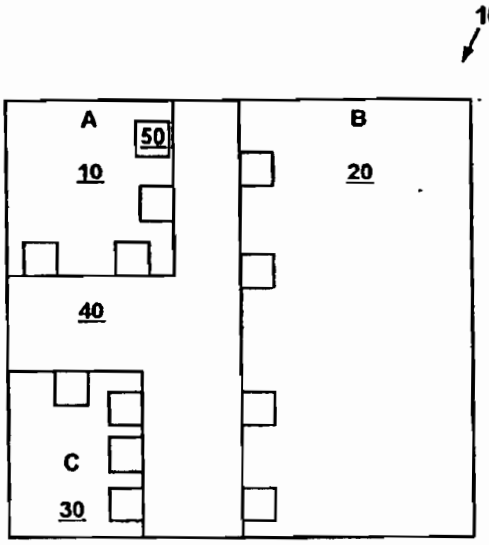


FIGURE 1
(PRIOR ART)

TITLE: OPTIMIZATION OF THE TOP LEVEL IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN
Inventor (s): Peter Dahl, Byron Dickinson, Margie Levine, Paul Rodman
USSN: 10/104,813 Attorney Docket #: RESH-001.CON1

Replacement Sheet 14/31

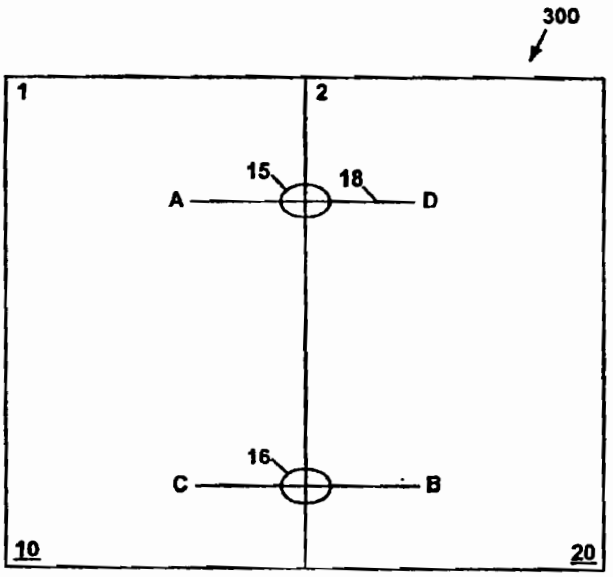


FIGURE 11A

REMARKS

Claims 2-41 were previously pending in this patent application. Claims 2-41 stand rejected. Herein, Claims 2, 12, 22, and 32 have been amended. Accordingly, after this Amendment and Response, Claims 2-41 remain pending in this patent application. Further examination and reconsideration in view of the arguments set forth below is respectfully requested.

SPECIFICATION

In the specification, several paragraphs have been amended to correct inadvertent typographical errors. No new matter was added.

DRAWINGS

Figure 1 was objected to because it was not labeled "PRIOR ART". In amended Figure 1, the label "PRIOR ART" has been added. No new matter was added.

In amended Figure 11A, a previously omitted horizontal line has been added. No new matter was added.

35 U.S.C. Section 102(e) Rejections

Claims 2, 12, 22, and 32 stand rejected under 35 U.S.C. 102(e) as being anticipated by Lavin et al., U.S. Patent No. 6,243,854 (hereafter Lavin). These rejections are respectfully traversed.

Independent Claim 2 recites:

> A method of pressing a top-level object into one or more blocks of a physical design, comprising:
> a) performing a shape operation to identify a portion of said top-level object that is within a boundary of a particular block, wherein said top-level object includes a plurality of properties, and wherein said shape operation ***preserves said properties*** associated with said portion of said top-level object; and
> b) ***moving*** said portion of said top-level object and said properties associated with said portion of said top-level object ***from a top-level netlist to a block-level netlist*** of said particular block ***so that said top-level object changes from a flattened state to a hierarchical state***. (emphasis added)

It is respectfully asserted that Lavin does not disclose the present invention as recited in Independent Claim 2. In particular, Lavin is directed to a method of providing the application programmer with additional or an extended set of modes for hierarchical processing. [Lavin; Col. 1, line 59 through Col. 2, line 7]. In Lavin, hierarchical relationships between shapes are specified to constrain the identification of pairs of shapes. Id. Each pair of shapes that satisfies the constraint (hierarchical relationship) is identified. Id. Examples of the constraints are "childcell", "parentcell", "samecell", and "peercell". [Lavin; Col. 2, lines 9-13]. A shape processing function (e.g., shape transformation,

measurement, predicate, etc.) is applied to these shapes. [Lavin; Col. 2, lines 5-7]. However, Lavin does not disclose that the shape operation preserves properties associated with the shapes.

Moreover, the Office Action at page 3 cites Col. 4, lines 58-64, of Lavin as disclosing "flattening or pressing or moving the top-level object(s) netlist (e.g., parent cell) into the block-level netlist (e.g., the resulting flattened netlist or shape or layer)". This citation fails to include the words "flattening", "pressing", "moving", and "netlist". Specifically, this citation simply discusses how the specification of hierarchical relationship (e.g., "childcell", "parentcell", "samecell", and "peercell") can be used for shape operations (e.g., intersection, distance, overlaps, spacing, etc.) that pertain to the relationships between shapes on different levels or layers, or among shapes on a single level or layer. However, Lavin does not disclose moving a portion of a top-level object and its properties from a top-level netlist to a block-level netlist so that the top-level object changes from a flattened state to a hierarchical state.

Additionally, the cited reference Ho (U.S. Patent 6,009,250) appropriately describes "flattening" as a process in which a design is flattened into the top level cell so as to remove hierarchical structure. [Ho; Col. 2, lines 15-21]. The flat representation is created by moving the geometry of leaf cells, e.g., those cells at the lowest level of the hierarchical tree, and the geometry of all intermediate

cells, upward through the hierarchical structure until all layout data and interconnect information resides at the top level. Id. Thus, "flattening" refers to moving objects from a lower level (e.g., block level) to the top level instead of referring to movement of objects from the top level to a lower level (e.g., block level) as stated in the Office Action at page 3.

Unlike Lavin, Independent Claim 2 is directed to a method of pressing a top-level object into one or more blocks of a physical design. The method includes performing a shape operation to identify a portion of the top-level object that is within a boundary of a particular block, wherein the top-level object includes a plurality of properties, and wherein the shape operation preserves the properties associated with the portion of the top-level object. Further, the method includes moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. While Lavin fails to disclose that the shape operation preserves properties associated with the shapes, Independent Claim 2 recites that the shape operation preserves the properties associated with the portion of the top-level object. Moreover, Lavin fails to disclose moving a portion of a top-level object and its properties from a top-level netlist to a block-level netlist so that the top-level object changes from a flattened state to a hierarchical state, as recited in Independent Claim 2. Lastly, while the Office Action at page 3

incorrectly refers to movement of objects from the top level to a lower level (e.g., block level) as "flattening" and cites Lavin as disclosing "flattening", Independent Claim 2 refers to movement of objects from the top level to a lower level (e.g., block level) as a hierarchical process and is supported by the description of "flattening" found in the cited reference Ho. Therefore, it is respectfully submitted that Independent Claim 2 is not anticipated by Lavin and is in condition for allowance.

With respect to Independent Claim 12, it is respectfully submitted that Independent Claim 12 recites similar limitations as in Independent Claim 2. In particular, the computer-readable medium of Independent Claim 12 comprises computer-executable instructions stored therein to perform a method of pressing a top-level object into one or more blocks of a physical design. The method includes performing a shape operation to identify a portion of the top-level object that is within a boundary of a particular block, wherein the top-level object includes a plurality of properties, and wherein the shape operation preserves the properties associated with the portion of the top-level object. Further, the method includes moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Therefore, Independent Claim 12 is allowable over Lavin for reasons discussed in connection with Independent Claim 2.

Independent Claim 22 recites:

A method of pressing a top-level object into one or more blocks of a physical design, comprising:
a) identifying a portion of said top-level object that is within a boundary of a particular block;
b) ***moving*** said portion of said top-level object ***from a top-level netlist to a block-level netlist*** of said particular block ***so that said top-level object changes from a flattened state to a hierarchical state***;
c) if said top-level object includes ***a press property***, performing a ***block-level placement*** for said particular block such that ***a block-level physical location of said portion of said top-level object is substantially equivalent to a top-level physical location of said portion of said top-level object***; and
d) if said top-level object ***does not*** include a press property, performing ***said block-level placement for said particular block without regard to said top-level physical location of said portion of said top-level object***. (emphasis added)

It is respectfully asserted that Lavin does not disclose the present invention as recited in Independent Claim 22. In particular, Lavin is directed to a method of providing the application programmer with additional or an extended set of modes for hierarchical processing. [Lavin; Col. 1, line 59 through Col. 2, line 7]. In Lavin, hierarchical relationships between shapes are specified to constrain the identification of pairs of shapes. Id. Each pair of shapes that satisfies the constraint (hierarchical relationship) is identified. Id. Examples of the constraints are "childcell", "parentcell", "samecell", and "peercell". [Lavin; Col. 2, lines 9-13]. A shape processing function (e.g., shape transformation,

measurement, predicate, etc.) is applied to these shapes. [Lavin; Col. 2, lines 5-7].

Further, the Office Action at page 3 cites (Col. 5, lines 16-55; Col. 2, line 64 to Col. 3, line 67; and Col. 2, lines 56-63) of Lavin to support "the performing based on the press property corresponds to the filter mechanism ... when applied using properties (i.e., parentcell, childcell, peercell) ... would result in step (c) and "samecell" property ... resulting in step (d)." First, Lavin neither discloses "block-level placement" nor uses the term "placement". Secondly, the filter mechanism refers to use of hierarchical relationships (e.g.,"childcell", "parentcell", "samecell", and "peercell") to identify pairs of shapes. For example, if the specified relationship is "peercell", then shape pair (sb, si) will be kept if and only if sb and si are shapes defined in peer cells. Further, the hierarchical relationships (e.g.,"childcell", "parentcell", "samecell", and "peercell") determine how the shape operation performed on the subject shape takes into consideration interactions with other shapes. This filter mechanism does not correspond to the press property since the filter mechanism does not impact performance of a block-level placement. For example, the filter mechanism does not determine whether performance of the block-level placement takes into account a top-level physical location of the portion of the top-level object moved from a top-level netlist to a block-level netlist.

In sum, Lavin does not disclose performing a block-level placement for a particular block that is dependent on whether a top-level object includes a press property. For example, Lavin fails to disclose performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object, if the top-level object includes the press property. Further, Lavin does not disclose performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object, if the top-level object does not include the press property.

Moreover, the Office Action at page 3 cites Col. 4, lines 58-64, of Lavin as disclosing "moving or flattening into the block level (i.e., child or same or peer cells or flattened layer)". This citation fails to include the words "flattening" and "moving". Specifically, this citation simply discusses how the specification of hierarchical relationship (e.g., "childcell", "parentcell", "samecell", and "peercell") can be used for shape operations (e.g., intersection, distance, overlaps, spacing, etc.) that pertain to the relationships between shapes on different levels or layers, or among shapes on a single level or layer. However, Lavin does not disclose moving a portion of a top-level object from a top-level netlist to a block-level netlist so that the top-level object changes from a flattened state to a hierarchical state.

Additionally, the cited reference Ho (U.S. Patent 6,009,250) appropriately describes "flattening" as a process in which a design is flattened into the top level cell so as to remove hierarchical structure. [Ho; Col. 2, lines 15-21]. The flat representation is created by moving the geometry of leaf cells, e.g., those cells at the lowest level of the hierarchical tree, and the geometry of all intermediate cells, upward through the hierarchical structure until all layout data and interconnect information resides at the top level. Id. Thus, "flattening" refers to moving objects from a lower level (e.g., block level) to the top level instead of referring to movement of objects from the top level to a lower level (e.g., block level) as stated in the Office Action at page 3.

Unlike Lavin, Independent Claim 22 is directed to a method of pressing a top-level object into one or more blocks of a physical design. The method includes moving a portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Further, the method includes if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object. Also, the method includes if the top-level object does not include a press property, performing the block-level placement for the

particular block without regard to the top-level physical location of the portion of the top-level object. While Lavin fails to disclose the press property and its impact on the performance of the block-level placement, Independent Claim 22 recites limitations directed to the press property and its impact on the performance of the block-level placement. Moreover, Lavin fails to disclose moving a portion of a top-level object from a top-level netlist to a block-level netlist so that the top-level object changes from a flattened state to a hierarchical state, as recited in Independent Claim 22. Lastly, while the Office Action at page 3 incorrectly refers to movement of objects from the top level to a lower level (e.g., block level) as "flattening" and cites Lavin as disclosing "flattening", Independent Claim 22 refers to movement of objects from the top level to a lower level (e.g., block level) as a hierarchical process and is supported by the description of "flattening" found in the cited reference Ho. Therefore, it is respectfully submitted that Independent Claim 22 is not anticipated by Lavin and is in condition for allowance.

With respect to Independent Claim 32, it is respectfully submitted that Independent Claim 32 recites similar limitations as in Independent Claim 22. In particular, the computer-readable medium of Independent Claim 32 comprises computer-executable instructions stored therein to perform a method of pressing a top-level object into one or more blocks of a physical design. The method includes moving a portion of the top-level object from a top-level netlist to a

block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Further, the method includes if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object. Also, the method includes if the top-level object does not include a press property, performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object. Therefore, Independent Claim 32 is allowable over Lavin for reasons discussed in connection with Independent Claim 22.

Claims 2, 6-9, 11-12, 16-19, 21-22, 26-29, 31-32, 36-39, and 41 stand rejected under 35 U.S.C. 102(e) as being anticipated by Ho et al., U.S. Patent No. 6,009,250 (hereafter Ho). These rejections are respectfully traversed.

It is respectfully asserted that Ho does not disclose the present invention as recited in Independent Claim 2. In particular, Ho is directed to a method of efficiently performing hierarchical design rules checks (DRC) and layout versus schematic (LVS) checks by identifying certain regions of the layout having overlapping designs positioned therein and further selectively flattening the layout database only for those identified overlapping areas. [Ho; Col. 3, lines 60-66].

Further, Ho describes "flattening" as a process in which a design is flattened into the top level cell so as to remove hierarchical structure. [Ho; Col. 2, lines 15-21]. The flat representation is created by moving the geometry of leaf cells, e.g., those cells at the lowest level of the hierarchical tree, and the geometry of all intermediate cells, upward through the hierarchical structure until all layout data and interconnect information resides at the top level. Id. Moreover, Ho fails to disclose a shape operation that preserves properties.

Further, Ho does not disclose performing a block-level placement for a particular block that is dependent on whether a top-level object includes a press property. For example, Ho fails to disclose performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object, if the top-level object includes the press property. Also, Ho does not disclose performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object, if the top-level object does not include the press property.

Unlike Ho, Independent Claim 2 is directed to a method of pressing a top-level object into one or more blocks of a physical design. The method includes performing a shape operation to identify a portion of the top-level object that is within a boundary of a particular block, wherein the top-level object includes a

plurality of properties, and wherein the shape operation preserves the properties associated with the portion of the top-level object. Further, the method includes moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. While Ho fails to disclose that the shape operation preserves properties associated with the shapes, Independent Claim 2 recites that the shape operation preserves the properties associated with the portion of the top-level object. Moreover, while Ho is directed to selective flattening (i.e., moving a lower-level object from a lower-level netlist to a top-level netlist), Independent Claim 2 is directed to moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Therefore, it is respectfully submitted that Independent Claim 2 is not anticipated by Ho and is in condition for allowance.

With respect to Independent Claim 12, it is respectfully submitted that Independent Claim 12 recites similar limitations as in Independent Claim 2. In particular, the computer-readable medium of Independent Claim 12 comprises computer-executable instructions stored therein to perform a method of pressing a top-level object into one or more blocks of a physical design. The method includes performing a shape operation to identify a portion of the top-level object

that is within a boundary of a particular block, wherein the top-level object includes a plurality of properties, and wherein the shape operation preserves the properties associated with the portion of the top-level object. Further, the method includes moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Therefore, Independent Claim 12 is allowable over Ho for reasons discussed in connection with Independent Claim 2.

Unlike Ho, Independent Claim 22 is directed to a method of pressing a top-level object into one or more blocks of a physical design. The method includes moving a portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Further, the method includes if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object. Also, the method includes if the top-level object does not include a press property, performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object. While Ho fails to disclose the press property and its impact on the performance of the block-level placement, Independent Claim 22 recites

limitations directed to the press property and its impact on the performance of the block-level placement. Moreover, Ho fails to disclose moving a portion of a top-level object from a top-level netlist to a block-level netlist so that the top-level object changes from a flattened state to a hierarchical state, as recited in Independent Claim 22. On the contrary, Ho is directed to selective flattening (i.e., moving a lower-level object from a lower-level netlist to a top-level netlist). Therefore, it is respectfully submitted that Independent Claim 22 is not anticipated by Ho and is in condition for allowance.

With respect to Independent Claim 32, it is respectfully submitted that Independent Claim 32 recites similar limitations as in Independent Claim 22. In particular, the computer-readable medium of Independent Claim 32 comprises computer-executable instructions stored therein to perform a method of pressing a top-level object into one or more blocks of a physical design. The method includes moving a portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state. Further, the method includes if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object. Also, the method includes if the top-level object does not include a press property, performing the block-level placement for the

particular block without regard to the top-level physical location of the portion of the top-level object. Therefore, Independent Claim 32 is allowable over Ho for reasons discussed in connection with Independent Claim 22.

Dependent Claims 6-9, 11, 16-19, 21, 26-29, 31, 36-39, and 41 are dependent on allowable Independent Claims 2, 12, 22, and 32, which are allowable over Ho. Hence, it is respectfully submitted that Dependent Claims 6-9, 11, 16-19, 21, 26-29, 31, 36-39, and 41 are patentable over Ho for the reasons discussed above.

35 U.S.C. Section 103(a) Rejections

Claims 3-5, 13-15, 23-25, and 33-35 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Ho et al., U.S. Patent No. 6,009,250 (hereafter Ho) in view of Bamji et al., U.S. Patent No. 5,604,680 (hereafter Bamji). These rejections are respectfully traversed.

Dependent Claims 3-5, 13-15, 23-25, and 33-35 are dependent on allowable Independent Claims 2, 12, 22, and 32, which are allowable over Ho. Moreover, Bamji does not disclose a method including performing a shape operation to identify a portion of the top-level object that is within a boundary of a particular block, wherein the top-level object includes a plurality of properties, and

wherein the shape operation preserves the properties associated with the portion of the top-level object, as recited in Claims 2 and 12. Further, Bamji does not disclose moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state, as recited in Claims 2 and 12.

Similarly, Bamji does not disclose a method including moving a portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state, as recited in Claims 22 and 32. Further, Bamji does not disclose if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object, as recited in Claims 22 and 32. Also, Bamji does not disclose if the top-level object does not include a press property, performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object, as recited in Claims 22 and 32.

Hence, it is respectfully submitted that Dependent Claims 3-5, 13-15, 23-25, and 33-35 are patentable over Ho and Bamji for the reasons discussed above.

Claims 10, 20, 30, and 40 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Ho et al., U.S. Patent No. 6,009,250 (hereafter Ho) in view of Levy et al., U.S. Patent No. 6,275,971 (hereafter Levy). These rejections are respectfully traversed.

Dependent Claims 10, 20, 30, and 40 are dependent on allowable Independent Claims 2, 12, 22, and 32, which are allowable over Ho. Moreover, Levy does not disclose a method including performing a shape operation to identify a portion of the top-level object that is within a boundary of a particular block, wherein the top-level object includes a plurality of properties, and wherein the shape operation preserves the properties associated with the portion of the top-level object, as recited in Claims 2 and 12. Further, Levy does not disclose moving the portion of the top-level object and the properties associated with the portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state, as recited in Claims 2 and 12.

Similarly, Levy does not disclose a method including moving a portion of the top-level object from a top-level netlist to a block-level netlist of the particular block so that the top-level object changes from a flattened state to a hierarchical state, as recited in Claims 22 and 32. Further, Levy does not disclose if the top-level object includes a press property, performing a block-level placement for the particular block such that a block-level physical location of the portion of the top-level object is substantially equivalent to a top-level physical location of the portion of the top-level object, as recited in Claims 22 and 32. Also, Levy does not disclose if the top-level object does not include a press property, performing the block-level placement for the particular block without regard to the top-level physical location of the portion of the top-level object, as recited in Claims 22 and 32.

Hence, it is respectfully submitted that Dependent Claims 10, 20, 30, and 40 are patentable over Ho and Levy for the reasons discussed above.

CONCLUSION

It is respectfully submitted that the above amendments, arguments and remarks overcome all rejections. For at least the above-presented reasons, it is respectfully submitted that all remaining claims (Claims 2-41) are now in condition for allowance.

The Examiner is urged to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present Application.

Please charge any additional fees or apply any credits to our PTO deposit account number: 23-0085.

Respectfully submitted,

WAGNER, MURABITO & HAO, LLP

Dated: 8/9/2004

Jose S. Garcia

Jose S. Garcia
Registration No. 43,628

Two North Market Street, Third Floor
San Jose, CA 95113
(408) 938-9060

Attachments

TITLE: OPTIMIZATION OF THE TOP LEVEL IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN
Inventor (s): Peter Dahl, Byron Dickinson, Margie Levine, Paul Rodman
USSN: 10/104,813 Attorney Docket #: RESH-001.CON1

14/31

Annotated Sheet Showing Changes

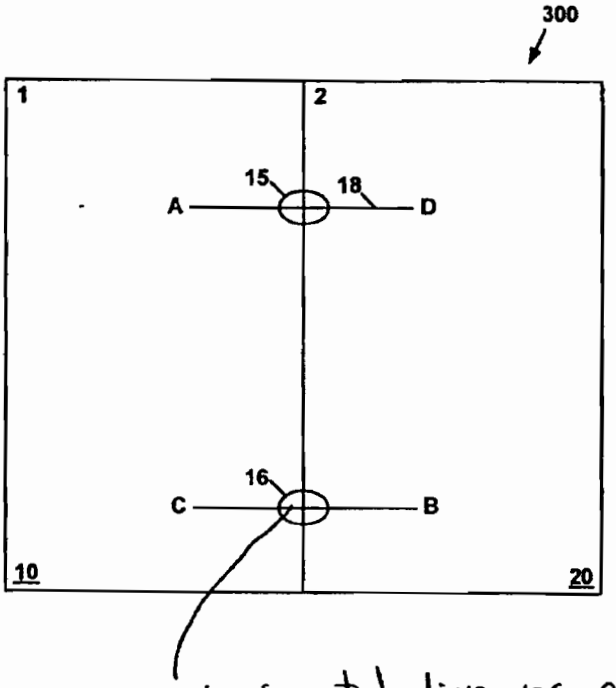


FIGURE 11A

TITLE: OPTIMIZATION OF THE TOP LEVEL IN ABUTTED-PIN HIERARCHICAL PHYSICAL DESIGN
Inventor (s): Peter Dahl, Byron Dickinson, Margie Levine, Paul Rodman
USSN: 10/104,813 Attorney Docket #: RESH-001.CON1

1/31

Annotated Sheet Showing Changes

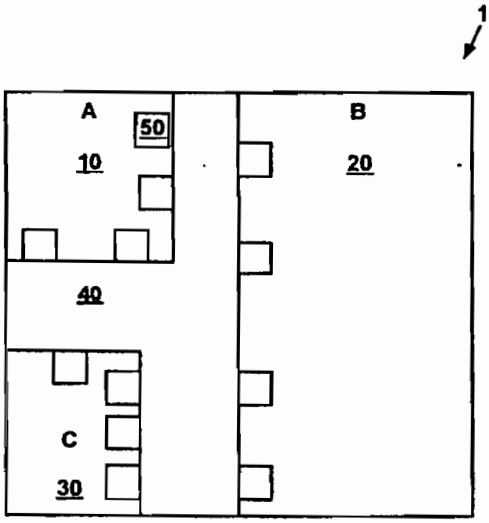


FIGURE 1
(PRIOR ART)

added "(Prior Art)"